Oracle® Application Server

Security Guide

Release 4.0.8.1

September 1999

Part No. A60116-03

Oracle Application Server Release 4.0.8.1 Security Guide

Part No. A60116-03

Contributing Authors: Sheryl Maring, Mukul Paithane, Raymond Ng, Alan Jones, Alka Srivastava

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the programs.

Contents

3 SSL with HTTP Requests

4 Directory Servers

5 Security for IIOP-based Applications: EJB, ECO/Java, and C++

6 Oracle Wallet Manager

Index

Preface

Audience

This guide is designed for administrators wishing to protect applications, static pages, and CGI scripts.

The Oracle Application Server Documentation Set

This table lists the Oracle Application Server documentation set.

Title of Book	Part No.
Oracle Application Server 4.0.8 Documentation Set	A66971-03
Oracle Application Server Overview and Glossary	A60115-03
Oracle Application Server Installation Guide for Sun SPARC Solaris 2.x	A58755-03
Oracle Application Server Installation Guide for Windows NT	A58756-03
Oracle Application Server Administration Guide	A60172-03
Oracle Application Server Security Guide	A60116-03
Oracle Application Server Performance and Tuning Guide	A60120-03
Oracle Application Server Developer's Guide: PL/SQL and ODBC Applications	A66958-02
Oracle Application Server Developer's Guide: JServlet Applications	A73043-01
Oracle Application Server Developer's Guide: LiveHTML and Perl Applications	A66960-02
Oracle Application Server Developer's Guide: EJB, ECO/Java and CORBA Applications	A69966-01
Oracle Application Server Developer's Guide: C++ CORBA Applications	A70039-01
Oracle Application Server PL/SQL Web Toolkit Reference	A60123-03
Oracle Application Server PL/SQL Web Toolkit Quick Reference	A60119-03

Title of Book	Part No.
Oracle Application Server JServlet Toolkit Reference	A73045-01
Oracle Application Server JServlet Toolkit Quick Reference	A73044-01
Oracle Application Server Cartridge Management Framework	A58703-03
Oracle Application Server 4.0.8.1 Release Notes	A66106-04

Conventions

This table lists the typographical conventions used in this manual.

Convention	Example	Explanation
bold	**oas.h** **owsctl** **wrbcfg** **www.oracle.com**	Identifies file names, utilities, processes, and URLs
italics	*file1*	Identifies a variable in text; replace this place holder with a specific value or string.
angle brackets	`<filename>`	Identifies a variable in code; replace this place holder with a specific value or string.
courier	`owsctl start wrb`	Text to be entered exactly as it appears. Also used for functions.
square brackets	`[-c string]`	Identifies an optional item.
	`[on|off]`	Identifies a choice of optional items, each separated by a vertical bar (|), any one option can be specified.
braces	`{yes|no}`	Identifies a choice of mandatory items, each separated by a vertical bar (|).
ellipses	n,...	Indicates that the preceding item can be repeated any number of times.

The term “Oracle Server” refers to the database server product from Oracle Corporation.

The term **“oracle**” refers to an executable or account by that name.

The term “*oracle*” refers to the owner of the Oracle software.

Technical Support Information

Oracle Global Support can be reached at the following numbers:

- In the USA: **Telephone: 1.650.506.1500**
- In Europe: **Telephone: +44 1344 860160**
- In Asia-Pacific: **Telephone: +61. 3 9246 0400**

Please prepare the following information before you call, using this page as a checklist:

❏ your CSI number (if applicable) or full contact details, including any special project information

❏ the complete release numbers of the Oracle Application Server and associated products

❏ the operating system name and version number

❏ details of error codes and numbers and descriptions. Please write these down as they occur. They are critical in helping WWCS to quickly resolve your problem.

❏ a full description of the issue, including:

  - **What -** What happened? For example, the command used and its result.
  - **When -**When did it happen? For example, during peak system load, or after a certain command, or after an operating system upgrade.
  - **Where -**Where did it happen? For example, on a particular system or within a certain procedure or table.
  - **Extent -** What is the extent of the problem? For example, production system unavailable, or moderate impact but increasing with time, or minimal impact and stable.

❏ Keep copies of any trace files, core dumps, and redo log files recorded at or near the time of the incident. WWCS may need these to further investigate your problem. For a list of trace and log files, see “Configuration and Log Files” in the *Administration Guide.*

For installation-related problems, please have the following additional information available:

❏ listings of the contents of $ORACLE_HOME (Unix) or %ORACLE_HOME% (NT) and any staging area, if used.

❏ installation logs (**install.log**, **sql.log**, **make.log**, and **os.log**) typically stored in the **$ORACLE_HOME/orainst** (Unix) or **%ORACLE_HOME%\orainst** (NT) directory.

Documentation Sales and Client Relations

In the United States:

- To order hardcopy documentation, call Documentation Sales: **1.800.252.0303.**
- For shipping inquiries, product exchanges, or returns, call Client Relations: **1.650.506.1500.**

In the United Kingdom:

- To order hardcopy documentation, call Oracle Direct Response: **+44 990 332200.**
- For shipping inquiries and upgrade requests, call Customer Relations: **+44 990 622300.**

Reader's Comment Form

Oracle Application Server Security Guide
Part No. A60116-03

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send your comments to:

Oracle Application Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

If you would like a reply, please provide your name, address, and telephone number below:

Thank you for helping us improve our documentation.

1

Overview of Security Features

Oracle Application Server provides several security schemes, or features, that enable you to control access to your applications and web pages. Protecting your applications is especially important if they contain sensitive data or if they are accessible through the Internet.

Contents

Terms

The following terms are used throughout this book:

Scheme In the context of this book, a *scheme* is defined as a security method or feature. There are authentication schemes, such as basic and digest, and there are access control schemes, such as IP and domain.

Authentication In the context of this book, the term *authentication* refers to the process of verifying that the user, or client, is who they say they are.

User Authentication

When access to an application or document is protected by a *user authentication scheme*, the server verifies that the client is who it claims to be. The user authentication schemes supported by Oracle Application Server are:

- Basic
- Basic_oracle
- Digest
- Crypt
- Certificate

Host Authentication

When access to an application or document is protected by a *host authentication scheme*, the server verifies the IP address or domain name of the client. The host authentication schemes supported by Oracle Application Server are:

- IP
- Domain

Access Control In the context of this book, the term *access control* refers to the granting (or denying) of permission to a user to access a resource (a file or application).

Auth Server In the context of this book, the term *Auth Server* is used to describe the part of Oracle Application Server that performs both *authentication* and *access control*. The Auth Server is comprised of two components, the broker and the provider.

Clients of the Auth Server include the dispatcher, the RM Proxy, the ICX (inter-cartridge exchange) service, and cartridges.

You can combine authentication schemes to protect applications. To access such applications, a user has to send the request from an authorized IP address or domain name, and also has to enter a valid username / password. For more information, see Chapter 2, "Authentication Schemes".

Broker The broker is the component of the Auth Server that directs traffic, sending security requests to the appropriate provider.

Provider The provider is the component of the Auth Server that implements the security schemes. There is one provider for each scheme, such as Basic or IP.

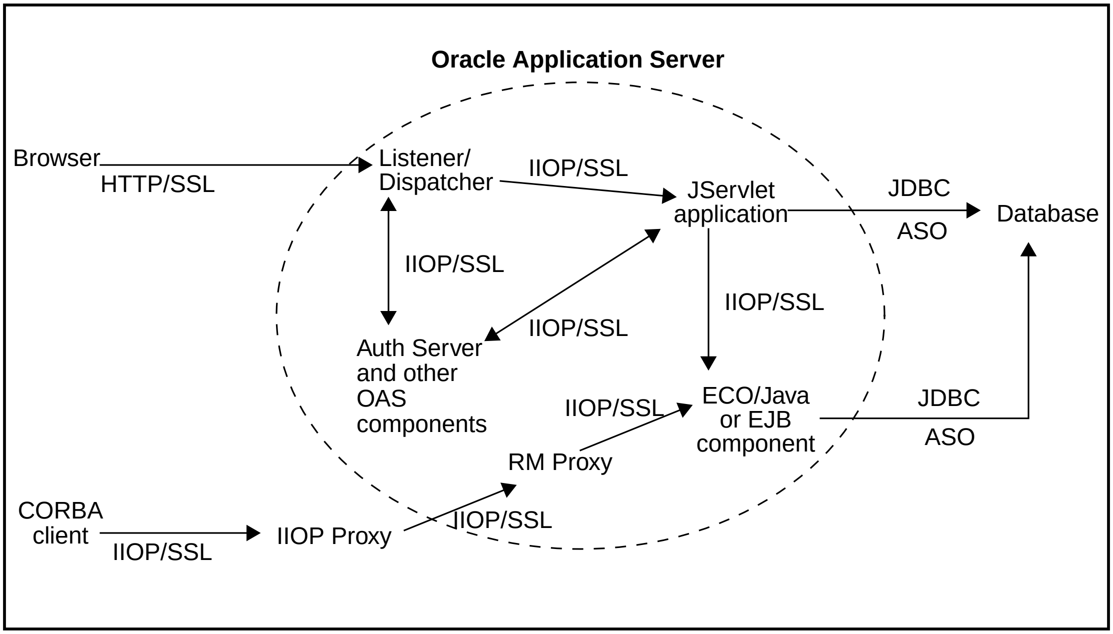
Encryption *Encryption* ia aprocess applied to plain text data that transforms it into scrambled data. Data transported between client and server can be encrypted if you enable SSL (Secure Sockets Layer) on the listener that receives the request.

Integrity *Integrity* is the insurance that the transmitted data has not been tampered with. Oracle Application Server supports integrity through SSL.

Security Architecture

The following figure shows communication paths between different components of Oracle Application Server and how each path is secured:

Figure 1–1 End-to-end security



When the Oracle Application Server components are distributed, they communicate over Common Object Request Broker Architecture (CORBA), an industry standard for cross-platform inter-operability between executable objects.

Following is a description of the components in Figure 1–1:

- **Browser -** This can be a typical browser like Netscape Navigator, a Web spider, a robot, or another Web-based application.
- **Listener -** The HTTP daemon (httpd) that is listening for requests for service.
- **Dispatcher -** The dispatcher handles HTTP requests that require cartridges. There is one dispatcher per listener. The dispatcher is composed of various components including the Auth Server, the Configuration Provider, the Logger, and so on.
- **Database -** This refers to an Oracle database. Web cartridges or CORBA components (ECOs, C++ cartridge, EJBs) can interface with an Oracle database via Oracle Call Interface (OCI) or other protocols. If the database is on a different machine, they communicate through Net8. For more information, see“Security Between Oracle Application Server and Database” on page 1-5.

- **RM Proxy -** The Resource Manager (RM) proxy is a Web Request Broker service that obtains object references to ECO/Java, EJB, and C++ objects and returns them to clients.

> **Note:** Oracle does not yet provide an IIOP Proxy. You need to use IIOP Proxies from third parties. Oracle Application Server has been tested with Gatekeeper, an IIOP Proxy from Visigenic. You can download a trial version of this product from the Visigenic website **http://www.visigenic.com**.

Security Between Oracle Application Server and Database

Communication between Oracle Application Server and Oracle databases is over Net8. Therefore, security between Oracle Application Server and an Oracle database is maintained by Net8, and not by Oracle Application Server.

Net8 Security

Net8 is a messaging layer that operates transparently across most common networking protocols or within the memory of a single machine. It enables an Oracle database to communicate with various other Oracle components, including Oracle Application Server, without having to be concerned with underlying network protocols or system architecture. Net8 has a set of its own encryption methods that can be set up to protect the message traffic.

Oracle's Advanced Security Option (ASO) provides an additional layer of security on top of the Net8 protocol for secure client to server communication. ASO integrates the Oracle database with a series of enterprise network services, including enterprise directory services, network encryption, single sign-on services, and access control services. Installation of these services makes the difference between basic client/server connectivity and enterprise client/server systems that are secure, manageable, configurable, and suitable for high-end deployment.

PL/SQL Cartridge Security

The PL/SQL cartridge is provided with the Oracle Application Server for the purpose of running PL/SQL applications in an Oracle database. The PL/SQL cartridge provides additional security for its clients that is beyond the security provided by the WRB API. A URL that invokes the PL/SQL cartridge specifies the DAD to be used for that request.

Database Access Descriptors (DADs)

Database Access Descriptors are used to control database access from the PL/SQL and JServlet cartridges. They can also, but need not, be used by any cartridge you write that accesses and Oracle database. For more information, see the DAD chapter in the *Oracle Application Server Administration Guide*.

If the DAD does not contain a username and password, or if the attempt to log on using the username and password in the DAD fails, the PL/SQL cartridge performs BASIC_NEW authentication. This means it will instruct Oracle Application Server to prompt for a username and password to log on to the database. The difference between this and Database Authentication is that, in this case, the Auth Server is not involved.

> **Note:** In this situation, the username and password are sent from the browser to the listener in the clear. If encryption is important, SSL Security should be enabled for all listeners invoking the PL/SQL cartridge.

Stored Procedure Access (Protecting Packages)

As shown above, the PL/SQL Cartridge enables a client to enter a URL that directly specifies a PL/SQL procedure to be executed by the Oracle database. This is desirable, but should be controlled. The only procedures that should be made available in this fashion are those intended to be entry points into your application. These procedures themselves should be able to call other procedures, without exposing the latter procedures to the public.

The PL/SQL Cartridge lets you enforce this by protecting packages. A protected package can only be executed by another stored procedure, not directly by a URL. By default, all the OWA utilities provided with the Oracle Application Server are protected. Other packages are by default unprotected.

Using the Oracle Application Server with Firewalls

A firewall is used to secure the Oracle Application Server across an Internet/ Intranet junction. Typically, a firewall may be placed:

- **in front of the listener -** In this case, the firewall provides access for HTTP and SSL for browser-to-listener communication.

- **behind Oracle Application Server, but before an Oracle database -** In this way, the data is kept secure and a firewall is used that allows secure Oracle Net8 protocols to pass through. Currently, the firewalls that permit Net8 traffic to pass through are as follows:

  - Trusted Information System's Gauntlet
  - Raptor

- **between the components of Oracle Application Server itself**

  Since Oracle Application Server has a distributed architecture (see "Security Architecture" on page 1-3), you can place a firewall between any two components that are configured to run in a single site. It is best to contact your firewall vendor before attempting this, however.

2

Authentication Schemes

Oracle Application Server uses the Auth Server for authentication of applications. This allows you to manage security at a high level across all cartridges or components.

Authentication schemes enable you to limit access to static pages, CGI scripts, and cartridges to authorized users. For an overview of security schemes and terms, see "Terms" on page 1-1.

Contents

Auth Server

The Auth Server component of Oracle Application Server handles security on behalf of dispatchers and cartridges/components. When a dispatcher receives an HTTP request requiring access to a protected virtual path, it contacts the Auth Server and waits for its response to determine whether the request can be filled.

If you are writing C cartridges, you can use WRB APIs to let the cartridge handle security. The cartridge submits a "authentication string" to the Auth Server for verification. The Auth Server functions the same whether it is invoked by a dispatcher or by a cartridge.

Under the Auth Server, passwords are stored in an encrypted format.

Components of the Auth Server

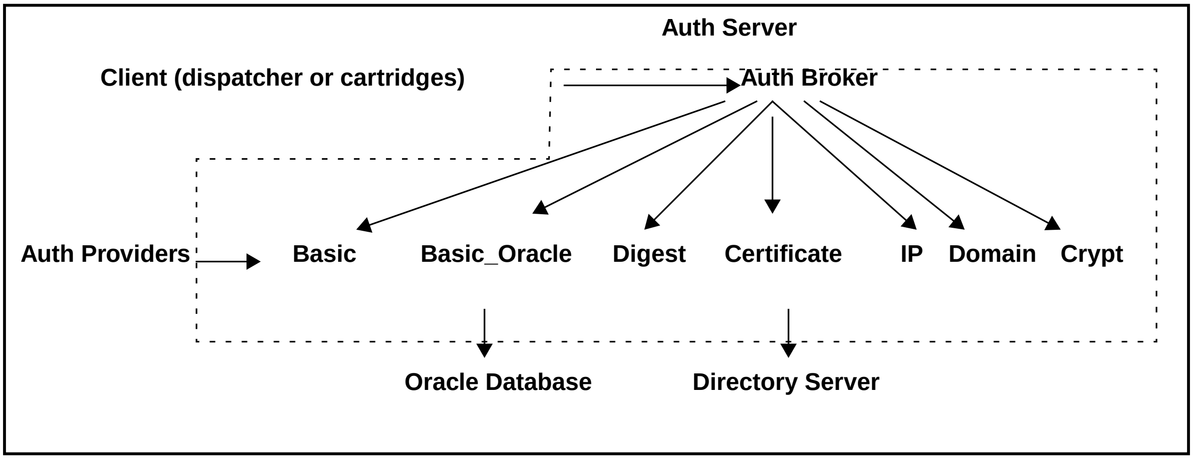
The Auth Server has two components: the Auth broker and the Auth providers:

- The **broker** receives security requests from Oracle Application Server components (dispatchers, RM Proxy, ICX, and wrks), and directs the request to the appropriate provider. The broker coordinates security requests; it does not perform the actual security check.
- **Providers** implement security schemes (basic, digest, basic_oracle, certificate, IP, or domain). For each scheme the broker recognizes, there is one provider.

When a client of the Auth Server (that is, a dispatcher or cartridge) sends a security request, it includes a "authentication string", which specifies a set of schemes and realms to validate. For more on authentication string, see "Access Control: Defining the Authentication String" on page 5-11. The broker forwards the authentication string to the appropriate provider to validate. The provider returns the validation result to the broker.

When it has received a response from the provider(s), the broker applies the AND/OR logic specified in the authentication string, and sends a message back to the dispatcher or cartridge that invoked it, indicating whether or not permission is granted.

Figure 2–1 Authorization brokers and providers



Auth Server Modes

The two components of the Auth Server (that is, the broker and the providers) can run in either of two modes: inmemory mode or ORB mode. The functionality is the same for either mode; the differences are in performance and opportunities for distribution.

The Inmemory Mode

In the “inmemory” mode, there is a separate instance of the broker / provider for each dispatcher and for each cartridge that invokes it. This increases the memory footprint of the Auth Server, but it speeds performance considerably.

In this mode, the Auth Server must be on the same machines as all modules that invoke it.

The ORB Mode

In the “ORB” mode, there is one instance of the broker / provider for all objects that will call it. This is slightly slower than the inmemory approach, but it requires less memory. More importantly, this approach enables you to have the Auth Server on a different machine than the objects that invoke it, enabling a distributed security architecture.

> **Note:** The providers for basic_oracle, crypt, and certificate schemes must be run in the ORB mode.

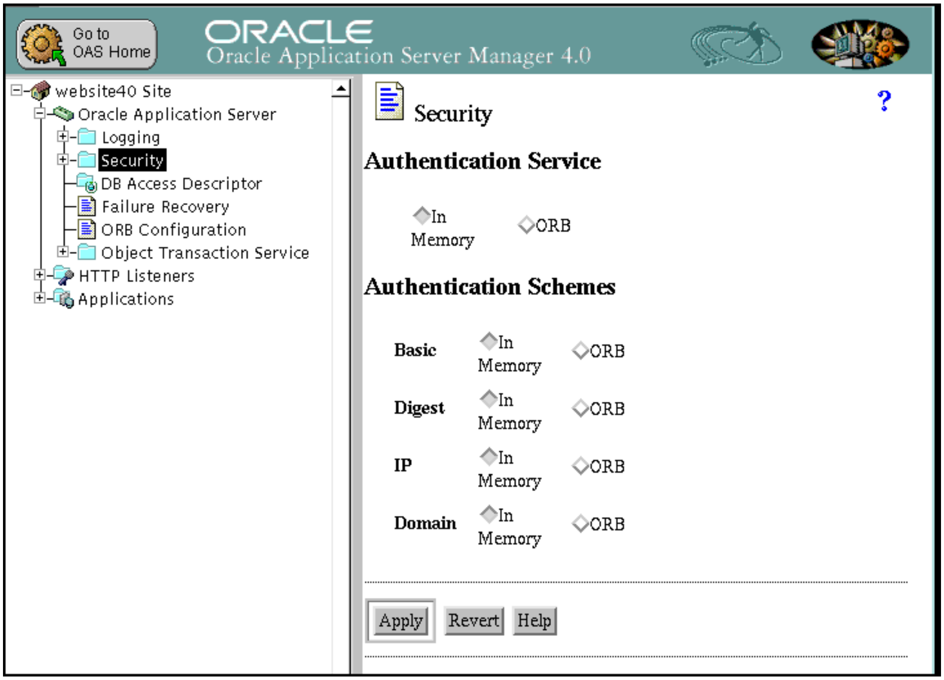
Setting the Mode

You set the mode of the Auth Server using the Security form in the Oracle Application Server Manager. To access the form:

1. From the Welcome page, select OAS Manager. (For instructions on connecting to the Welcome page, see the *Oracle Application Server Adminstration Guide.*)
2. Expand the Site folder.
3. Expand the Oracle Application Server folder.
4. Click on Security (without expanding the folder).

   The Security form appears:

Figure 2–2 Security form



Security

Authentication Service

◇ In Memory ◇ ORB

Authentication Schemes

Basic	◇ In Memory	◇ ORB
Digest	◇ In Memory	◇ ORB
IP	◇ In Memory	◇ ORB
Domain	◇ In Memory	◇ ORB

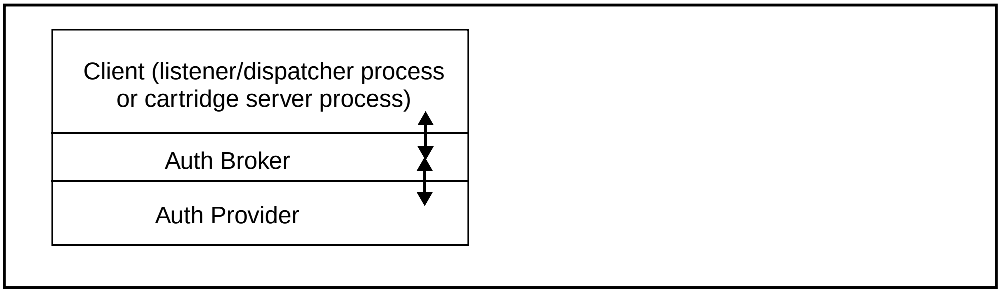
On the Security form, the "Authentication Service" field refers to the Auth broker, and "Authentication Schemes" refers to the Auth providers.

Mode Combinations

You can set different modes for the broker and the providers. You can define the following configurations:

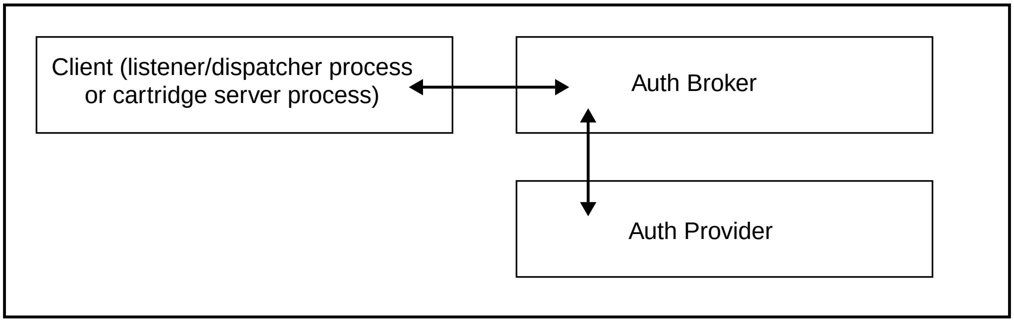
Broker inmemory, provider inmemory In this configuration, you get the best performance, but also the largest processes because the broker and providers are linked into the client processes.

Figure 2–3 Broker and provider inmemory


Client (listener/dispatcher process
or cartridge server process)
Auth Broker
Auth Provider

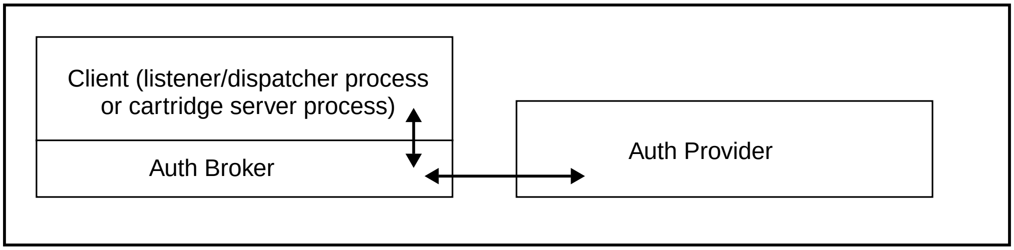
Broker ORB, provider ORB In this configuration, you get the best possibilities for distribution, since the brokers and the providers are separate processes. The trade-off is slower performance, since the communication is between processes.

Figure 2–4 Broker and provider ORB


Client (listener/dispatcher process
or cartridge server process)
Auth Broker
Auth Provider

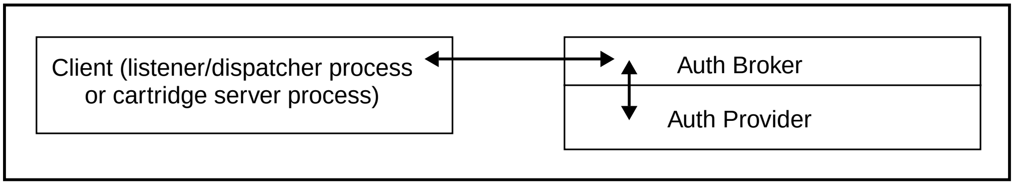
Broker inmemory, provider ORB In this configuration, the broker is linked into the client process. The broker communicates with the providers over the ORB.

Figure 2–5 Broker inmemory, provider ORB


Client (listener/dispatcher process
or cartridge server process)
Auth Broker
Auth Provider

Broker ORB, provider inmemory In this configuration, the broker and the providers are linked together. Clients communicate with the broker over the ORB.

Figure 2–6 Broker ORB, provider inmemory


Client (listener/dispatcher process
or cartridge server process)
Auth Broker
Auth Provider

User and Host Authentication

You can use authentication schemes to protect your site. Authentication schemes can be divided in two categories:

- User Authentication Schemes
- Host Authentication Schemes

User Authentication Schemes

Oracle Application Server supports the following user authentication schemes:

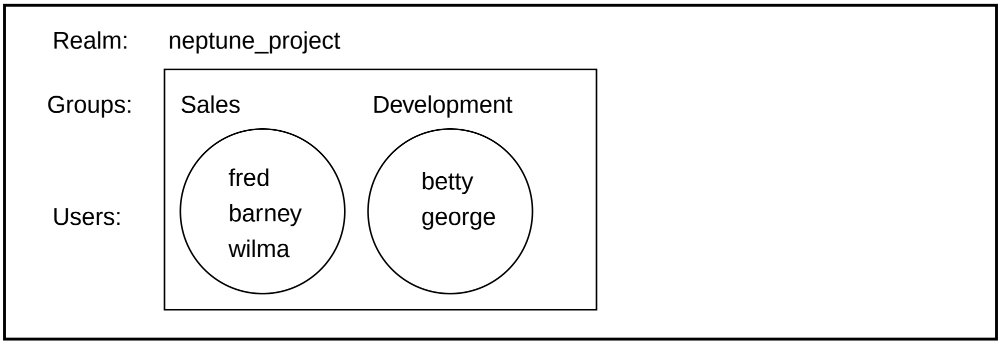
- basic
- digest
- basic_oracle
- crypt

A user authentication scheme performs security by prompting the user for a username/password combination. You can protect static HTML, CGI scripts, cartridges, or CORBA components by associating the virtual path in which they are stored with a security scheme. When a user attempts to access the protected virtual path, the dispatcher will return HTTP status 401 to the browser, which in turn prompts the user with a username/password dialog box.

Basic and Digest

Basic and digest authentication schemes are part of the HTTP 1.1 specification. In the basic and digest schemes, you define realms, groups, and users. Users belong to one or more groups, which belong to one or more realms. A user must belong to at least one group, and a group must belong to at least one realm. The following figure shows five usernames belonging to two groups, which belong to one realm ("neptune_project").

Figure 2–7 Usernames, groups, and realms for basic and digest schemes



Usernames, groups, and realms are defined separately for the basic and digest schemes. Usernames, groups, and realms defined in the basic scheme cannot be used in the digest scheme, and vice versa.

The realms, groups, and users that you define are saved in Oracle Application Server's configuration file. In the file, passwords for the users are stored in an encrypted format.

To protect a virtual path using the basic or digest scheme, you associate the virtual path with a realm. When a user requests the virtual path, the user is prompted to enter a username and password. For the authentication to succeed, the user has to enter a username in one of the groups in the realm. In the figure above, any of the five users have the ability to access the data.

Password Security In the basic scheme, the username/password are sent Base64 encoded to the server.

The digest scheme is based on a "challenge/response" paradigm. The server generates a nonce value and uses it as a challenge. The client sends a MD5 checksum of the username, the password, the nonce value, the HTTP method, and the requested URI. The password is never sent in clear.

The digest scheme is considerably better than the basic scheme. However, only a few browsers currently support digest authentication. Spyglass Mosaic does, but Netscape Navigator and Microsoft Internet Explorer do not. If digest authentication is specified for a page, and a user with a browser that does not support digest authentication tries to access that page, the listener automatically reverts to basic authentication.

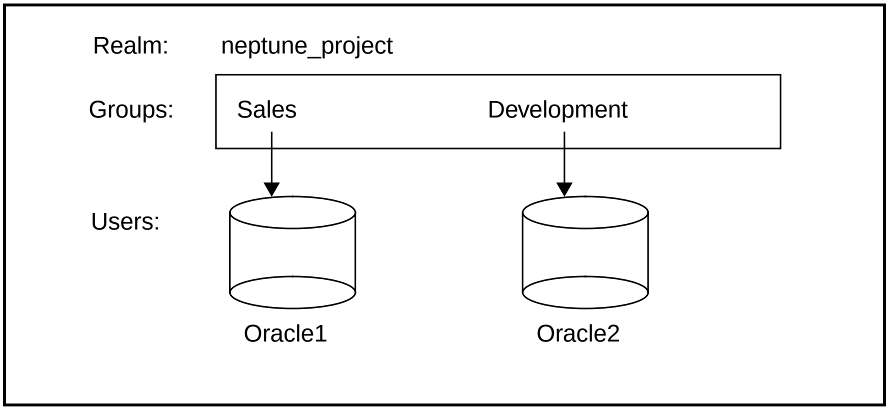
You should treat digest as effectively equivalent to basic for the majority of users and take precautions accordingly. When password encryption is a concern, you should use SSL instead of digest authentication. See Chapter 3, "SSL with HTTP Requests" for details.

Basic_Oracle (or Database) Scheme

In the basic_oracle (or database) authentication scheme, you also define realms, groups, and users. This is similar to the basic or digest schemes. The difference is that instead of listing the users in each group, you associate a group with an Oracle database. The authentication succeeds if the username/password provided by the user can be used to log into an Oracle database associated with the groups in the realm. If the login succeeds, the user is then logged out of the database.

In the following figure, if a virtual path is protected by the "neptune_project" realm, a user can access the virtual path if he or she enters a valid username/password for either of the associated databases.

Figure 2–8 Usernames, groups, and realms for basic_oracle scheme



This scheme enables you to manage only one set of users. By managing the users in an Oracle database, you can also determine who has access to the protected virtual paths.

In addition, a group may specify a database role, which further restricts the authorized users by allowing only those database users that have the privilege to assume that role to be authenticated.

UNIX Crypt Authentication

Use the crypt authentication feature to control access to cartridges using UNIX-style encryption of passwords.

Crypt reads usernames and encrypted passwords from a user-defined file, for example **/tmp/group1.file**, as opposed to the **wrb.app** file. The file is a static file and must reside on the same node that provides the crypt security. This means the file should reside on the same node as the `wrbahsrv`. By default this is the primary node; however, if other `wrbahsrv` processes are running on other nodes, you need to copy the file to all other nodes.

The syntax of the file is *username*:*encrypted_password*. The fields are separated by a colon. For example:

```
User1:sg340tsdgTRS
User2:gvio0M8IesIY
```

The recommended way of getting the encrypted password is from the **/etc/passwd** file.

If you do not have access to the **/etc/passwd** file, you can call the `crypt()` function (see the man page for *crypt(3C)* for details on the function) to generate the encrypted password. A drawback of generating encrypted passwords manually is that you have to know each user's password. The following C program shows how you can use the `crypt()` function.

```
#include <stdio.h>
#include <crypt.h>

main()
{
    char *encrypted_passwd;
    char user[512], clearPassword[512];

    printf("Enter User:");
    gets(user);

    printf("Enter ClearText Password:");
    gets(clearPassword);

    encrypted_passwd = crypt(clearPassword, "ab");

    printf("%s:%s\n", user , encrypted_passwd);
}
```

When you run the program, you get the following output (*generate_password* is the name of the program):

```
prompt> generate_password
Enter User:fred
Enter ClearText Password:sample1
fred:abHCYPDdd7pug
```

The crypt provider is only accessible in ORB mode.

Host Authentication Schemes

Host authentication schemes enable you to list machines or domains that can or cannot access a virtual path. When a user requests a virtual path protected by an host authentication scheme, the server checks the IP address or domain of the machine sending the request and compares it to the authentication string. The server fulfils the request only if the client machine has permission to access the page or application.

You can use host authentication schemes to protect static HTML pages, CGI applications, and cartridges/components.

Oracle Application Server supports two host authentication schemes:

- IP
- domain

Both allow you to you specify who has or does not have access. In IP host authentication, you authenticate based on the machine's IP address. In the domain host authentication, you authenticate based on the machine's domain name.

In both schemes, a + sign in front of an IP address or domain name specifies that the machine has access to the virtual path, and a - sign specifies that the machine does not have access. You can use the * wildcard character in the IP address or domain name.

Examples:

Table 2–1 Examples of host authentication schemes

Example	Description
+144.25.10.100	Allow access to machine with that IP address.
-144.66.44.111	Deny access to machine with that IP address.
-140.84.4.*	Deny access to machines whose IP address begins with 140.84.4.
-140.84.4.* +*	Allow access to all addresses except those in 140.84.4.*
+test23.us.oracle.com	Allow access to machine with that host name.
+144.25.10.*	Allow access to machines whose IP address begins with 144.25.10.
-*.acme.com	Deny access to machines from the acme.com domain.

In both schemes, you define an host authentication group and associate the group with one or more patterns. For example, you can define a group called "mycompany" and associate it with a pattern such as "`+*.mycompany.com, -*.xyz.com`". Patterns are separated by a comma. Searching stops when a match is found.

There is an implicit `-*` at the end of each list. This means that if the user machine's IP or domain name does not match a pattern in the list, then it is disallowed.

Configuring Authentication Schemes

You use the Oracle Application Server Manager to configure authentication schemes. Establishing security for your site is divided into two tasks:

- Task 1: Define Users, Groups, and Realms
- Task 2: Associate Realms or Groups with Virtual Paths

Task 1: Define Users, Groups, and Realms

In order to protect a site, you must establish who can and who cannot access the site. This is accomplished by creating a list (or lists) of users who will be allowed or denied access to material in your site. These users are then assigned to one or more groups. These groups are assigned to one or more realms. Security is performed by associating these groups or realms to a virtual path.

The steps for setting up these groups and realms are slightly different, depending on whether you are securing CGI scripts, or static HTML; or you are securing cartridges, components, or applications:

- Protecting CGI Scripts and Static HTML
- Protecting Cartridges, Components and Applications

Protecting CGI Scripts and Static HTML

1. Connect to the Welcome page.
2. Click on OAS Manager.
3. Expand the Site folder.
4. Expand the HTTP Listeners folder.
5. Expand the folder of the listener you wish to configure.
6. Expand the Security folder.

   The available security schemes appear:

   - Basic
   - Digest
   - IP
   - Domain
   - SSL

7. Go to “Select a Security Scheme” below.

Protecting Cartridges, Components and Applications

1. Connect to the Welcome page.
2. Click on OAS Manager.
3. Expand the Site folder.
4. Expand the Oracle Application Server folder.
5. Expand the Security folder.

   The available security schemes appear:

   - Basic
   - Digest
   - IP
   - Domain
   - Basic_Oracle
   - Crypt
6. Go to “Select a Security Scheme” below.

Select a Security Scheme

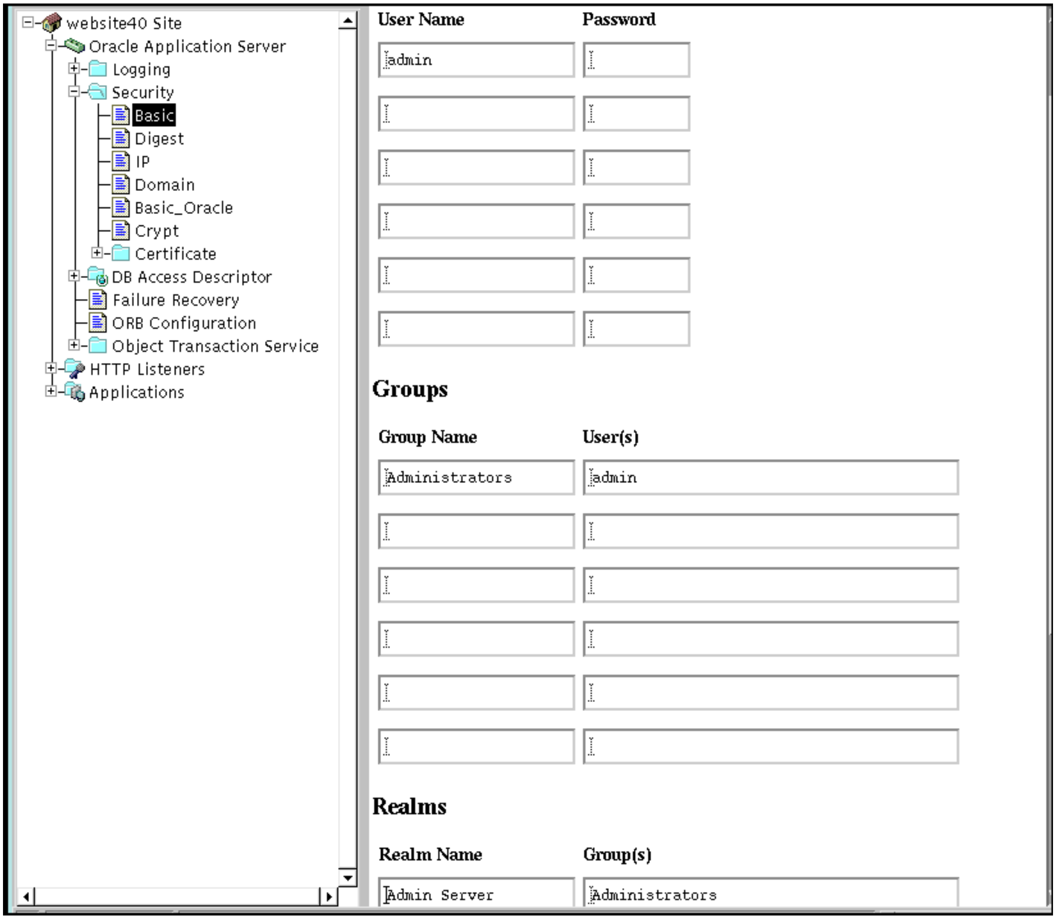
Your options are:

- Basic and Digest Authentication
- Basic_Oracle Authentication
- Crypt Authentication
- IP host authentication
- Domain host authentication

Basic and Digest Authentication

1. Select the authentication scheme (basic or digest) that you want. This displays the Basic or Digest form. The two forms are similar.

Figure 2–9 Basic form



User Name	Password
admin	

Groups

Group Name	User(s)
Administrators	admin

Realms

Realm Name	Group(s)
Admin Server	Administrators

2. In the Basic or Digest form:

   - User Name: enter the usernames of those to be allowed access
   - Password: enter the passwords for the usernames
   - Group Name: enter one or more group names
   - User(s): enter the users that belong to the group. The user names are coma-separated.

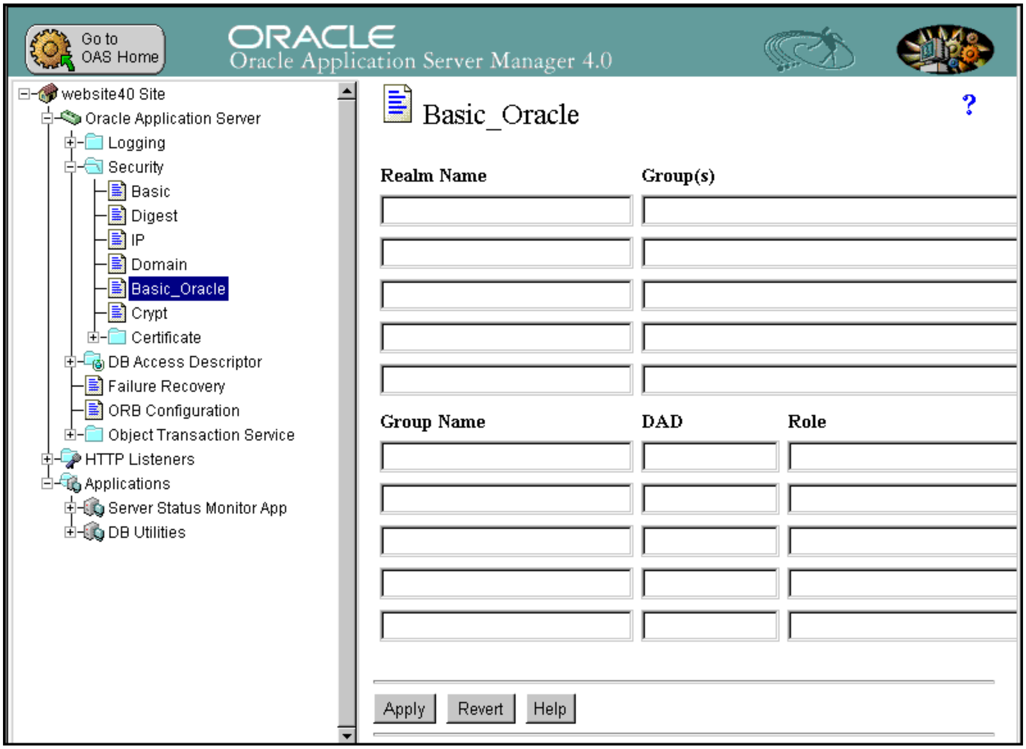
- Realm Name: enter a realm name, which you will later associate with virtual paths you want to protect. The realm name will be displayed in the username/password dialog.
- Group(s): enter the groups that belong to the realm. The group names are space-separated.
- Click Apply.

3. Restart the components in Oracle Application Server for the new configuration to take effect.

Basic_Oracle Authentication

1. Select Basic_Oracle. This displays the Basic_Oracle form.

Figure 2–10 Basic_Oracle form



2. In the Basic_Oracle form:

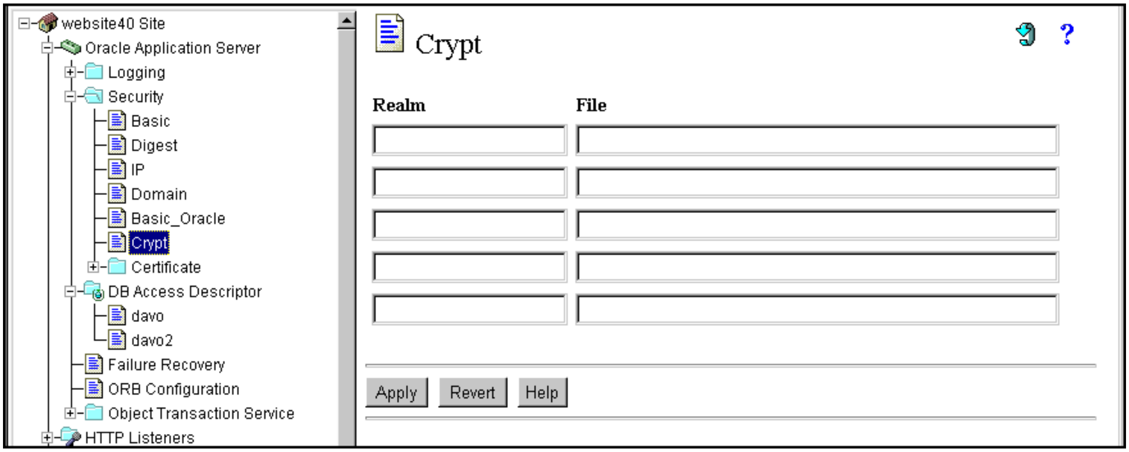
- Realm Name: enter the name of a realm, which you will later associate with virtual paths you want to protect. The realm name is also displayed in the username/password dialog box.
- Group(s): enter the groups that belong to the realm. The groups are defined in Group Name below. The group names are comma-separated.
- Group Name: enter the name of a group.
- DAD: enter the name of a database access descriptor (DAD) for this group. A DAD defines an Oracle database. To create DADs, select DB Access Descriptor, which is located under Oracle Application Server in the navigational tree.
- Role: enter the name of a database role. This parameter is optional. Use this field to specify that only a subset of database users (those who have the privilege to assume the role) are authorized. Roles allows you to divide users into groups within the database.

3. Click Apply.
4. Restart the components in Oracle Application Server for the new configuration to take effect.

Crypt Authentication When you use Crypt, you define realm name and file name.

1. Select Crypt.

Figure 2–11 Crypt form



2. In the Crypt form:

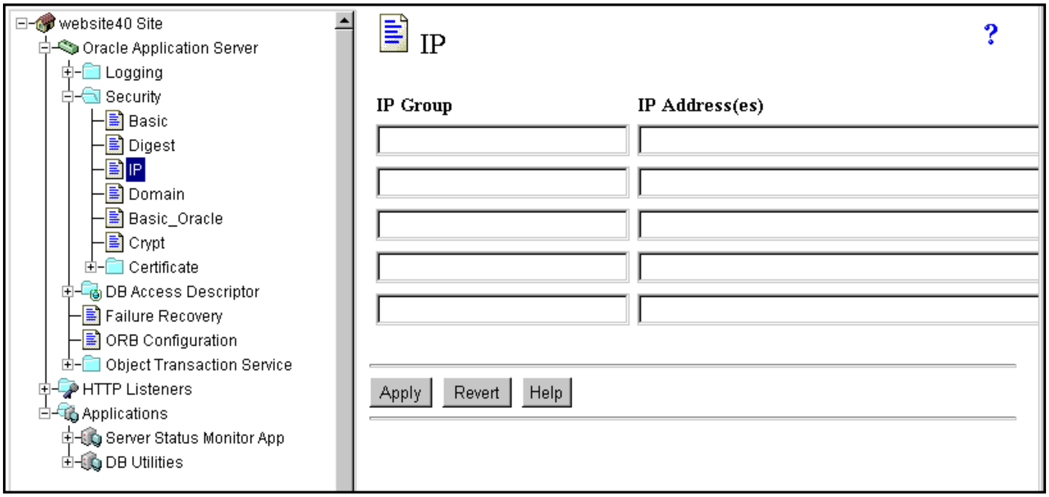
   - Realm Name: enter a realm name, which you will later associate with virtual paths you want to protect.
   - File Name: enter the files that belong to the realm.
   - Click Apply.

3. Restart the components in Oracle Application Server for the new configuration to take effect.

IP host authentication

1. Select IP. This displays the IP form.

Figure 2–12 IP form



2. In the IP form:

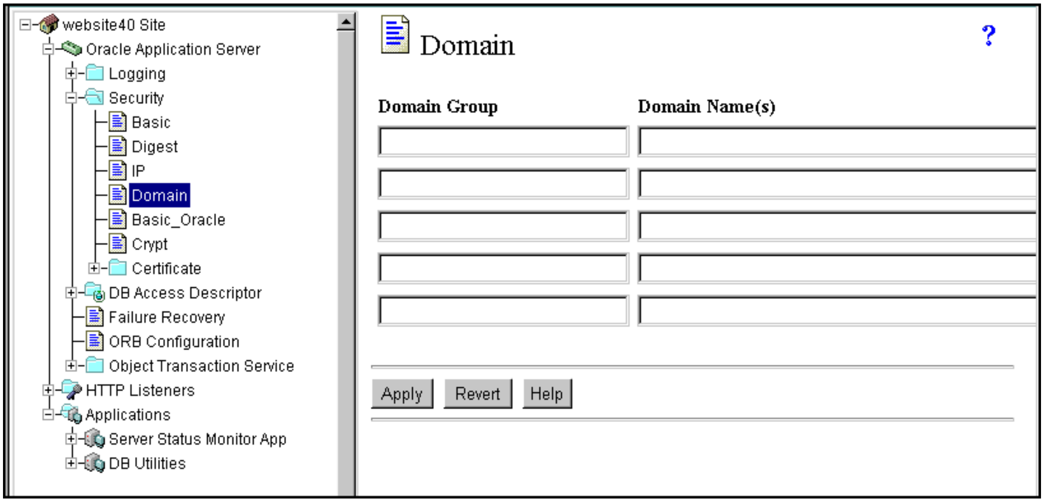
   - IP group: enter the IP group. This name is used to identify the IP host authentication list.
   - IP addresses: enter the list of IP addresses to restrict or allow access. Multiple addresses are separated by commas. Any IP address not included in an IP group is implicitly denied access to files and directories protected by that group.

3. Click Apply.
4. Restart the components in Oracle Application Server for the new configuration to take effect.

Domain host authentication

1. Select Domain. This displays the Domain form.

Figure 2–13 Domain form



2. In the Domain form:
   - Domain group: enter the domain group. This name is used to identify the host authentication list.
   - Domain names: enter the list of domain names to restrict or allow access. Multiple addresses are separated by spaces. Any domain names not included is implicitly denied access to files and directories protected by that group.
3. Click Apply.
4. Restart the components in Oracle Application Server for the new configuration to take effect.

Task 2: Associate Realms or Groups with Virtual Paths

After you have defined authentication realms or host authentication groups, you associate the realms or groups with virtual paths you want to protect.

You can protect individual HTML pages or all the pages in a directory. If you have a set of pages that you want to protect, you might want to place them in the same directory (or the same root directory), so that you can use the * wildcard character to protect all HTML files in a directory recursively.

The following table shows some examples:

Table 2–2 Protecting paths

Virtual path	Description
/files/index.html	Protect **index.html** only.
/files/*	Protect all the files in the physical path associated with **/files**. Subdirectories under **/files** are not protected.
/files/	Protect all the files in the physical path associated with **/files**. Subdirectories under **/files** are protected.
/files/x_*	Protect files whose names begin with x_.

In addition to saving you from typing individual filenames, using the * wildcard character prevents users from gaining access by bypassing the “authentication” page. Using the * character ensures that all pages require authentication.

The steps for associating groups or realms with virtual paths are slightly different, depending on what you are securing. If you are protecting CGI scripts or static HTML, security is performed at the listener level. If you are securing cartridges, components, or applications, security is performed at the Oracle Application Server level. See the following section depending on your situation:

- Protecting CGI Scripts and Static HTML
- Protecting Cartridges, Components and Applications

Protecting CGI Scripts and Static HTML

To protect virtual paths for static pages and CGI scripts:

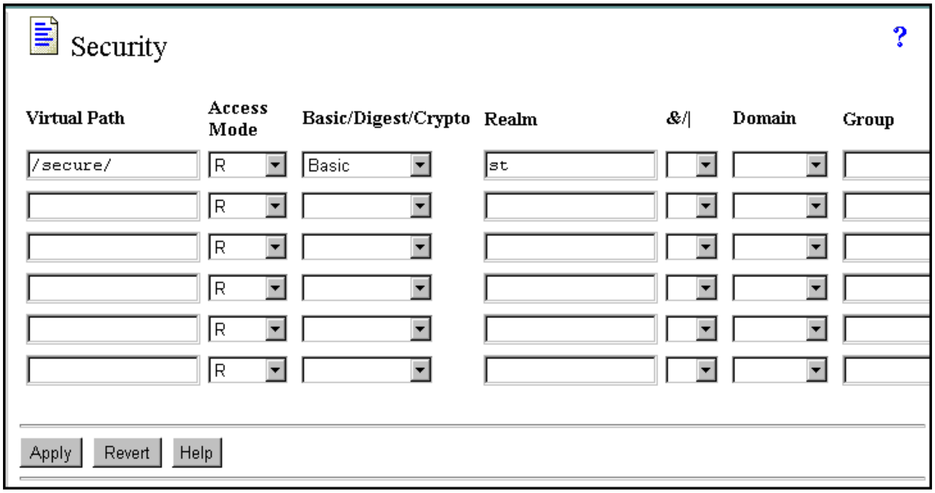
1. Connect to the Welcome page and click on OAS Manager.
2. Expand the Site folder.
3. Expand the HTTP Listeners folder.

4. Expand the listener with the virtual path you want to protect.

5. Click on Security (without expanding the folder). If necessary, enter the username and password. By default, these are the same as for the OAS Manager.

   The Security form appears:

Figure 2–14 Listener Security form



Virtual Path	Access Mode	Basic/Digest/Crypto	Realm	&/|	Domain	Group
/secure/	R	Basic	st			
	R					
	R					
	R					
	R					
	R					

6. In the Security form:

   - Virtual Path: enter the path you want to protect. You can specify protection for all files in the directory by using wildcard characters.
   - Access Mode: select how the files can be accessed. It is a combination of the following access modes:

Table 2–3 Access mode for files

Access Mode	How files or directories can be accessed
R	Using GET, HEAD, or POST, although the specific method may depend on the type of resource being accessed
W	Using PUT
D	Using DELETE

- Basic/Digest/Crypto: Whether to apply basic, digest, or Crypto SSL to the specified files or directories. If you use this field to specify an authentication scheme, you must specify a realm in the Realm field.
- Realm: enter the name of a realm to associate with the virtual path. The realm must exist in the selected scheme.
- For a virtual path, you can associate it with an authentication scheme and/or an host authentication scheme. The “&/|” column indicates whether the user has to satisfy both schemes or just one scheme.
- Domain/IP: select the host authentication scheme you want
- Group: enter the name of an IP or domain group to associate with the virtual path. The group must exist in the selected scheme.

7. Click Apply.
8. Restart the components in Oracle Application Server for the new configuration to take effect.

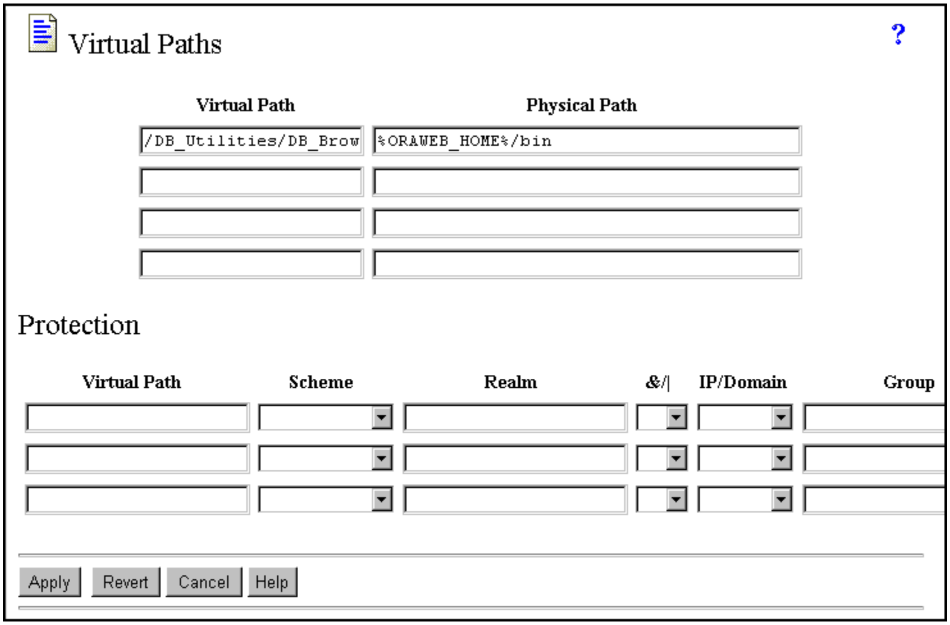
Protecting Cartridges, Components and Applications

To protect virtual paths for cartridges:

1. From the Welcome page, click on OAS Manager.
2. Click the Site folder.
3. Expand the Applications folder.
4. Expand the folder of the Application you wish to configure.
5. Expand the Cartridges folder.
6. Expand the folder of the cartridge you wish to configure.
7. Expand the Configuration folder.
8. Click on Virtual Path.

   The Virtual Paths form appears:

Figure 2–15 Virtual Paths form



Virtual Paths

Virtual Path	Physical Path
/DB_Utilities/DB_Brow	%ORAWEB_HOME%/bin

Protection

Virtual Path	Scheme	Realm	&/|	IP/Domain	Group

9. In the Virtual Paths form:

   - At the top of the form, you can create a new virtual path by entering the virtual path and the physical path.
   - In the “Protection” section, enter the virtual path you want to protect.
   - Scheme: select Basic, Digest, or Basic_Oracle, Certificate, or Crypt.
   - Realm: enter the name of a realm to associate with the virtual path. The realm must exist in the selected scheme.
   - & / |: select & if the user has to satisfy both schemes. Select **|** if the user has to satisfy only one scheme.
   - Domain / IP: select the host authentication scheme you want.
   - Group: enter the name of an IP or domain group to associate with the virtual path. The group must exist in the selected scheme.

10. Click Apply.

11. Restart the components in Oracle Application Server for the new configuration to take effect.

Implications of Multiple Listeners

The Oracle Application Server provides support for multiple listeners on a network, each having equal access to the same set of cartridges or CORBA components. To the extent that you handle your authorization at the listener level, your Oracle Application Server installation is only as secure as your least secure listener.

- Any application protected at the dispatcher level is only protected from users entering the listener associated with that dispatcher. All other listeners must have similar security in order for the protection to be meaningful.
- An SSL secured site may only be meaningful if other listeners which are part of that installation are similarly secured. If you are aware of this situation, you can use it deliberately. For example, you can use SSL listeners when users will be prompted for database passwords, but not when the passwords will be taken from the DAD. See “PL/SQL Cartridge Security”.
- Any new listeners added to the system must be immediately secured with the same (or better) security than is available at the other listeners.
- A new listener can be seamlessly added to the system by any unauthorized user as long as that user has the shared key of the installation available. This makes it more critical that the shared key is kept secure.

3

SSL with HTTP Requests

This chapter explains how to set up Oracle Application Server to impose public key cryptography on the communication between the listener and the browser using Secure Sockets Layer (SSL). Oracle Application Server supports SSL v2 and v3.

Contents

Overview

SSL is a transport protocol that provides for privacy, integrity, and authentication. These terms are defined below:

Privacy Messages are protected from being read by unintended recipients. SSL supports privacy by encrypting messages between clients and servers. Messages are encrypted using public keys, and can be decrypted only by the private key associated with the public key.

Integrity Messages are protected from being altered. If altered, messages cannot be decrypted correctly. SSL supports integrity by using the MD5 and SHA1 message-digest algorithm.

Authentication SSL verifies that the user is who it claims to be by using certificates. When a browser connects to a server, the server presents its certificate. The browser then either accepts that certificate automatically (if it is included in the “site certificates” in the browser) or prompts the user to decide whether to accept the certificate. The certificate allows the user to be sure that the server is who it is claiming to be.

In addition to the server being authenticated, the server can request that the client be authenticated. The client must have its own certificate, and it presents the certificate to the server when prompted. Client authentication is not required as part of the SSL protocol, and both Oracle Application Server and Netscape listeners allow you to configure this.

Clients and servers can get certificates from certificate authorities (CAs). CAs are entities that issue certificates to individuals or companies only after verifying the individual or company. An example of a CA is VeriSign, Inc.

Certificates

URLs that require SSL use “https” in place of “http”. The listener must be configured to accept an SSL connection, and the system administrator must have a certificate for the listener (usually obtained from Verisign or generated by the Oracle Security Server).

SSL works at the transport level, which is one level below the application level. This means that SSL can encrypt and decrypt messages before they are handled by application-level protocols such as Telnet, FTP, and HTTP.

In addition to providing for encryption, SSL also provides for authentication. When the browser connects to the server, the server presents its certificate. The browser then either accepts that certificate (if it is included in the “site certificates” in the browser) or prompts the user to decide whether to accept the certificate. The certificate allows the user to be sure that the server is who it is claiming to be. However, the simple fact that a server has a certificate does not mean the user must or should permit access.

In addition to server being authenticated, the server can request that the client be authenticated. The client must have obtained its own certificate and present that certificate to the server when prompted. Note, client authentication is not required as part of the SSL protocol and both the Oracle Application Server and Netscape listeners allow this to be configured.

SSL encryption and authentication can be used for static HTML pages, CGI scripts, and cartridges. If a request for a cartridge uses SSL, then the resulting HTML page generated by the cartridge is encrypted before it is sent.

For more information about SSL, see the following sites:

- For the SSL version 3.0 specification, see **http://home.netscape.com/eng/ssl3/ssl-toc.html**.

Access Control List

In addition to the site certificate, Oracle Application Server maintains an access control list (ACL), which lists the privileges for each component. For Oracle Application Server components, including ECO / Java objects, the ACL grants them the execute privilege, which enables them to execute methods on each other.

To protect the site, you can neither change this setting nor can you extend this privilege to objects that are not components of Oracle Application Server. This prevents malicious external CORBA objects from getting a list of objects in your site and invoking methods on them (for example, shutting them down or crashing your site).

You should install only ECO / Java objects that you trust on your site because all ECO / Java objects have the execute privilege.

If you are running a distributed site, where the components run on different machines, you have to install the site certificate and the ACL each machine.

Supported SSL Cipher Suites

The Oracle HTTP listener supports the following SSL cipher suites:

- 40bit https listener
  - SSL_RSA_EXPORT_WITH_RC4_40_MD5
  - SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
- 128bit https listener (in addition to the 40bit ciphers)
  - SSL_RSA_WITH_DES_CBC_SHA
  - SSL_RSA_WITH_3DES_EDE_CBC_SHA
  - SSL_RSA_WITH_RCA_128_SHA

- SSL_RSA_WITH_RCA_128_MD5

Configuring Oracle Application Server to Use SSL

To set up SSL on your site, you must perform the following tasks:

Server side:

- Task 1: Generate a Certificate Request
- Task 2: Submit a Request for a Certificate
- Task 3: Install Your Certificate
- Task 4: Set up a Port to Accept SSL Requests

Client side (optional):

- Submit a request to a CA for a certificate.
- Install the certificate.
- Ensure that the URL uses “https” instead of “http”. For example:

  https://www.mycompany.com/secure.htm

Task 1: Generate a Certificate Request

> **Note:** If you are upgrading from a version of Oracle Application Server prior to 4.0.8, you **must** upgrade your private key. See “Upgrading an Existing Private Key File” on page 3-16.

To use SSL, you need certificates. Oracle Application Server supports the X.509 v2 and X.509 v3 certificate formats.

Oracle Application Server provides **genreq**, a utility located in **$ORAWEB_HOME/bin/** that you can use to generate a new certificate request or upgrade an existing private key. The utility generates a key pair (private and public keys), and writes the public key information into a file called (by default) **certreq.pkc** which you can send to a CA to get a certificate. It also generates a file that contains the corresponding private key.

Note: The **genreq** utility requires that values be provided for all the name subcomponents of the Distinguished Name (DN): non NULL values have to be provided for all elements such as Country (C), Organization (O), Organization Unit (OU), State (ST), Locality (L), Common Name (CN).

To create a new certificate request using **genreq**:

1. Change directory to **$ORAWEB_HOME/bin**.

   ```
   prompt> cd $ORAWEB_HOME/bin
   ```

2. Type **genreq** to start the utility.

   ```
   prompt> genreq
   ```

   ```
   Certificate Request Generator
   ```

   G - Generate key pair and certificate request
   U - Upgrade an old private key file (pre 4.0.8) to new format
   Q - Quit

   ```
   > Enter choice:
   ```

3. Type `G` to begin creating a certificate request.

   ```
   G <RET>
   ```

4. Type a password for generating a private key. This must be at least eight characters.

   ```
   > Enter password (at least 8 characters) for creating a
   private key or press <Return> to cancel:
   ```

   ```
   ********<RET>
   ```

   (The password will not be displayed.)

5. Retype the password for confirmation.

   ```
   > Confirm the password:
   ```

   ```
   ********<RET>
   ```

6. Choose the public exponent you want to use in generating the key pair. The only two recognized exponents are 3 and 65537, commonly called Fermat 4 or F4. The default is F4.

```
> Specify the public exponent used to generate key pair
[F4]:
<RET>
```

7. Enter the size in bits of the modulus you want to use in generating the key pair. For the version of **genreq** sold in the United States of America and Canada, the size may be from 1 to 1024. The default size is 768 bits. A modulus size between 768-1024 is recommended.

   For versions of **genreq** sold outside the USA, the maximum (and default) modulus size is 512 bits.

   ```
   > Enter modulus size [768]:

   <RET>
   ```

8. Choose one of three methods for generating a random seed to use in generating the key pair:

   ```
   > Choose how you want to generate a random seed for the key
   pair.

   F - Random file
   K - Random key sequences
   B - Both
   Enter choice:
   ```

   F **- genreq** prompts you to enter the full pathname of a file in your local file system. This can be any file that is at least 256 bytes in size, does not contain any secret information, and has contents that cannot easily be guessed.

   K **- genreq** prompts you to enter random keystrokes. **genreq** uses the variation in time between keystrokes to generate the seed. Do not use the keyboard's autorepeat capability, and do not wait longer than two seconds between keystrokes. **genreq** prompts you when you have typed enough keystrokes. You must delete any unused characters typed after this prompt.

   B **- genreq** prompts you to enter both a file name and random keystrokes. This option is recommended.

   ```
   B <RET>

   > Enter the name of file to use as a source of random
   seed information:

   /home/test <RET>

   Type random characters (about 400) until you hear a beep:
   ```

```
........................................................
........................................................
.....................................................
```

```
> Stop typing
```

```
> Accept? [Y]
```

```
<RET>
```

9. Enter the name of a file in which to store your server’s distinguished name. You can choose the default, or enter a filename with a **.der** extension. **genreq** creates this file in the current directory.

```
> Enter the name of the distinguished name file
[servname.der]:
```

```
<RET>
```

10. Enter the name of a file in which to store your server’s private key. You can choose the default, or enter a filename with a **.der** extension. **genreq** creates this file in the current directory. Do not specify a location that is accessible via NFS or other network protocol.

```
> Enter the name of the private key file [privkey.der]:
```

```
<RET>
```

11. Enter the name of a file in which to store the certificate request. You can choose the default, or enter any filename with a **.pkc** extension.

> Enter the name of the certificate request file [**certreq.pkc**]:

```
<RET>
```

12. Enter the requested identification information for your organization:

    a. `Enter your Common Name (1 to 64 chars):`

    ```
    www.acme.com <RET>
    ```

    Common Name - the fully qualified host name of your organization’s Internet point of presence as defined by the Domain Name Service (for example, **www.oracle.com**).

    b. `Enter your Organization Unit Name (1 to 64 chars):`

    ```
    Advanced Development <RET>
    ```

Organizational Unit Name - the name of the group, division, or other unit of your organization responsible for your Internet presence, or an informal or shortened name for your organization.

c. `Enter your Organization Name (1 to 64 chars):`

```
Acme <RET>
```

Organization Name - the legal name of your company. Most CAs require you to verify this name by providing legal documents, such as a business license.

d. `Enter your Locality Name (1 to 128 chars):`

```
Redwood Shores <RET>
```

Locality - the city where your organization is located.

e.
```
> Enter your State or Province (1 to 128 chars)
[default: California]:
```

```
<RET>
```

State or Province - the full name of the state or province where your organization is located. VeriSign does not accept abbreviations.

f. `Enter your Country Name (2 chars) [default:US]:`

```
<RET>
```

Country - the two-character ISO-format abbreviation for the country where your organization is located. The country code for the United States is "US".

g. `Enter your Web Master's name (1 to 64 chars):`

```
John Doe <RET>
```

Web Master's Name - the name of the person managing Oracle Application Server.

h. `Enter your Web Master's E-mail address (1 to 128 chars):`

```
jdoe@acme.com <RET>
```

Email Address - the email address where the CA can contact the web master.

i.
```
Enter the name and version number of application for
which you are getting the certificate (1 to 64 chars)
[Oracle Web Listener]:
```

```
<RET>
```

Name and Version - The name and version number of the application for which you are getting the certificate.

13. When you have entered all the requested information, **genreq** responds with `Thank you`, and processes the data you have entered. When it is finished, it returns you to the main menu. (Type `Q` to exit the program.)

```
Thank you.
```

```
G - Generate key pair and certificate request
U - Upgrade an old private key file (pre 4.0.8) to new for-
mat
Q - Quit
> Enter choice:
```

```
Q <RET>
```

The **genreq** utility generates the following files:

- **certreq.pkc -** This is the file you send to the CA.
- **servname.der -** This file is your server's distinguished name.
- **privkey.der -** This file contains your private key. You do not send this file to anyone.

To request a certificate, email the **certreq.pkc** file to a CA. The certification process can take from a few days to several weeks. The more organized and complete your paperwork, the better your chances are for quick certification.

Task 2: Submit a Request for a Certificate

Once you have completed the steps above to generate a certificate request, you can submit a request for a certificate to a third-party CA such as Verisign Inc. (**http://www.verisign.com**).

Task 3: Install Your Certificate

When you receive your certificate, you use the SSL form in the Oracle Application Server Manager to install it. You can install one certificate per port. If you have a listener that listens on multiple ports, you can install a certificate for each port for that listener.

1. Use your email reader to save the message from the CA containing the certificate to a file.

2. Use a text editor to remove the header information before the BEGIN CERTIFICATE line and the footer information after the END CERTIFICATE line. Do **not** delete or alter the BEGIN CERTIFICATE and END CERTIFICATE lines themselves.

1. From the Welcome page, click on OAS Manager.

2. Expand the Site name.

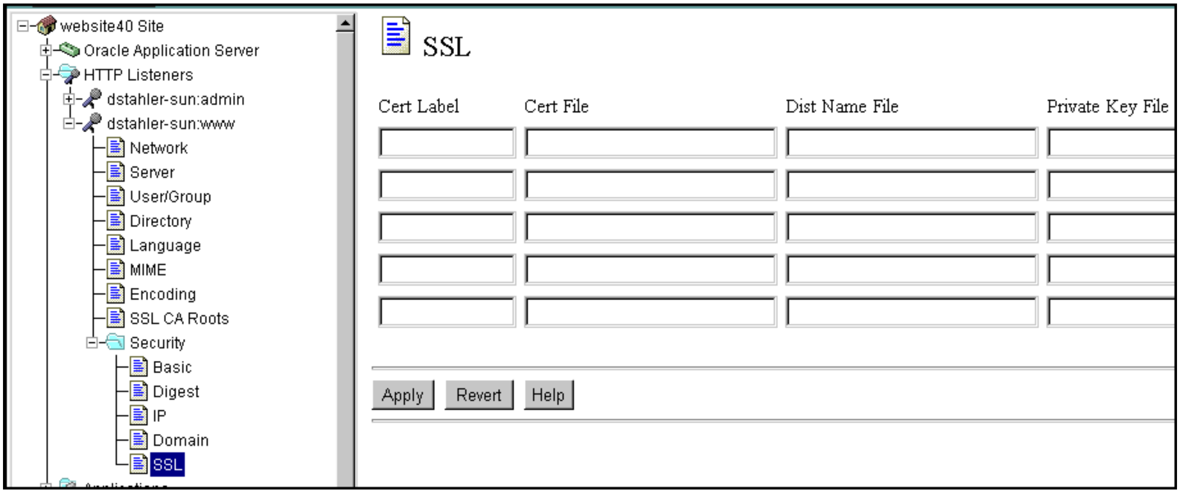
3. Expand HTTP Listeners.

4. Click “+” next to the listener that will be accepting SSL requests.

5. Click “+” next to Security.

6. Click SSL to display the SSL form.

Figure 3–1 SSL form



7. In the SSL form, fill in the fields:

Cert Label Enter the label to the certificate that you received from your CA. This is a name that you create to identify the certificate.

Cert File Enter the full path to the certificate file that you received from your CA. The file must be in one of the following formats:

- ASCII 64 encoded DER format (**.der** extension)
- PEM certificate approval message format (**.pem** extension)
- PKCS certificate approval message format (**.pkc** extension)

Dist Name File Enter the full path to a file that contains the server's distinguished name information. If **genreq** is used to generate a certificate request, you can use the *servname***.der** file generated by **genreq**. The file must be in one of the following formats:

- ASCII 64 encoded DER format (**.der** extension)
- PEM certificate approval message format (**.pem** extension)

Private Key File Enter the full path to a file that contains an encrypted version of your RSA private key. If **genreq** is used to generate a certificate request, you can use the **privkey.der** file generated by **genreq**. The private key file must be in one of the following formats:

- ASCII 64 encoded DER format (**.der** extension)
- PEM certificate approval message format (**.pem** extension)

CA Dir Enter the full path to the directory where additional certificate files from your CA can be found.

This feature is not currently supported. You can enter a directory name such as **/tmp**.

CRL Dir Enter the full path to the directory where additional Certificate Revocation List (CRL) files can be found.

This feature is not currently supported. You can enter a directory name such as **/tmp**.

8. Click Apply.

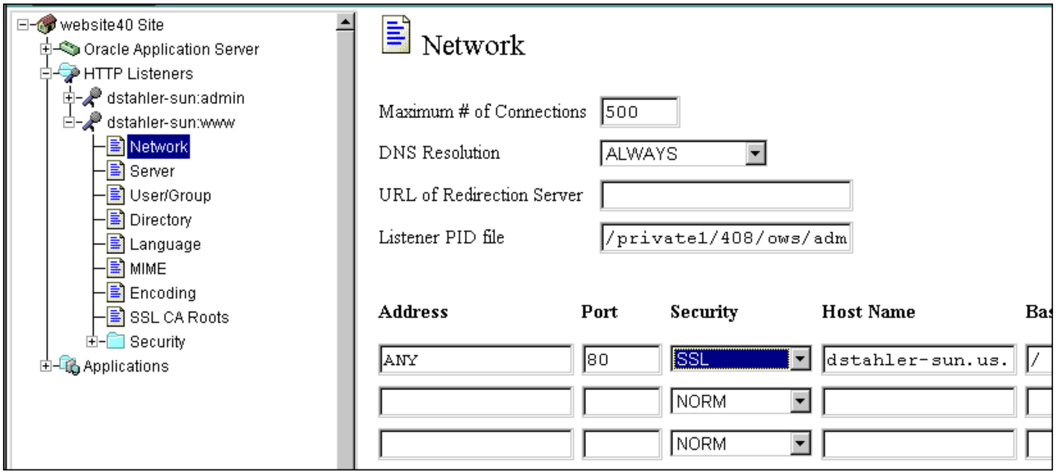
Task 4: Set up a Port to Accept SSL Requests

You must enable SSL on one or more ports for one or more listeners. Port 443 is the default port for SSL connections. This means that if a user enters a URL using https (instead of http), the user does not have to enter the port number after the server name for listeners listening at port 443 for SSL. Port 443 is analogous to port 80 for regular non-secure HTTP connections.

To set up a port to accept SSL requests:

1. From the Welcome page, click on OAS Manager.
2. Expand the Site folder.
3. Expand the HTTP Listeners folder.
4. Expand the folder of the listener that will be accepting SSL requests.
5. Click on Network to display the Network form.

Figure 3–2 Network form



Network

Maximum # of Connections 500

DNS Resolution ALWAYS

URL of Redirection Server

Listener PID file /private1/408/ows/adm

Address	Port	Security	Host Name	Bas
ANY	80	SSL	dstahler-sun.us.	/
		NORM		
		NORM		

6. In the Network form, the following fields apply to SSL (see the *Administration Guide* for information on the other fields on this form):

   Port - Enter the port number for the listener. The port number is usually 443, which is the default port for SSL connections.

   Security - Select one of the SSL options:

   - **SSL -** Clients using SSL 2.0 and / or SSL 3.0 are supported. Listener will understand initial SSLV2 Hello messages, but will try to negotiate up to SSL 3.0. If the client does not support SSL 3.0, then further communication will proceed using SSL 2.0.

- **SSL_V2 -** Communication is possible only using SSL 2.0. No communication takes place if SSL2 is not checked in your browser configuration.
- **SSL_V3 -** Communication is possible only using SSL 3.0. No communication takes place if SSL3 is not checked in your browser configuration.
- **SSL_V3_V2H -** Forces an SSL 3.0 connection, but allow a meaningful error when SSL 2.0-only clients attempt communication. The listener will understand initial SSL2.0 Hello messages, but will try to negotiate up to SSL 3.0. If the client does not support SSL3.0, then further communication will fail.

Authentication - Specify if the listener authenticates a client over an SSL port. This parameter is used for clients that are using SSL 3.0 only. Listeners do not authenticate clients that are using SSL 2.0. Select one of the following:

- **NONE -** Oracle Application Server does not authenticate clients.
- **OPT** (Optional) **-** Oracle Application Server requests the client for a certificate, but does not require one. If the client elects to send a certificate, however, it must be valid.
- **REQ** (Required) **-** Oracle Application Server requires clients to send their certificates. Oracle Application Server aborts the SSL handshake if a client does not send a certificate or if the CA of the issurer is not Trusted. See "Managing Trust Points (SSL CA Roots)", below.

Certificate Label

Enter the label of a certificate for the type of security selected in the Security field. You can have only one certificate label per port.

7. Click Apply.

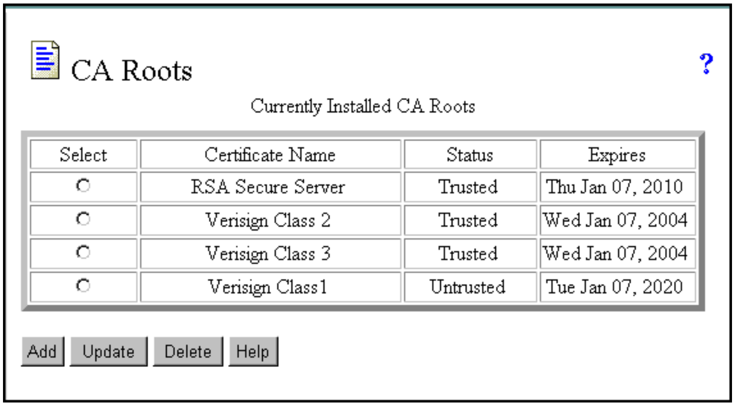
Managing Trust Points (SSL CA Roots)

A trustpoint is a third party entity that is qualified with a level of trust, and is used when an identity is being validated for access to a listener. Trustpoints are also referred to as CA (Certification Authority) Roots. Use the CA Roots administration form to manage CA Roots. To access this form:

1. From the Welcome page, click on OAS Manager.
2. Expand the Site folder.
3. Expand the HTTP Listeners folder.
4. Expand a listener (for example, www).

The CA Roots form appears.

Figure 3–3 CA Roots form



CA Roots

Currently Installed CA Roots

Select	Certificate Name	Status	Expires
○	RSA Secure Server	Trusted	Thu Jan 07, 2010
○	Verisign Class 2	Trusted	Wed Jan 07, 2004
○	Verisign Class 3	Trusted	Wed Jan 07, 2004
○	Verisign Class1	Untrusted	Tue Jan 07, 2020

CA Roots Form

This form allows you to control which root certificates your website will accept. By default, Oracle Application Server trusts the four listed SSL CA roots:

- VeriSign Class 1 Primary CA (VeriSign, Inc.)
- VeriSign Class 2 Primary CA (VeriSign, Inc.)
- VeriSign Class 3 Primary CA (VeriSign, Inc.)
- RSA Secure Server Certification Authority (RSA Data Security, Inc.)

This section describes the form’s fields.

Status: Trusted vs. Untrusted

A “Trusted” CA root means that users who authenticate with that particular CA root will be granted access. If a CA root is “Untrusted”, those who use that CA root will be denied access.

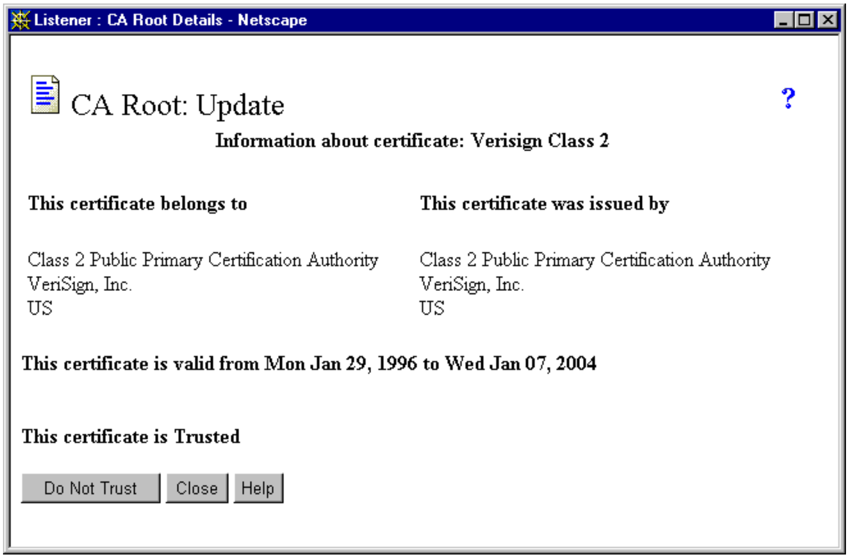
Updating a Certificate

To change the Status of a CA root:

1. Select the certificate.
2. Click Update.

3. A window appears with details about that particular certificate.

Figure 3–4 Update CA Root form



CA Root: Update

Information about certificate: Verisign Class 2

This certificate belongs to	**This certificate was issued by**
Class 2 Public Primary Certification Authority VeriSign, Inc. US	Class 2 Public Primary Certification Authority VeriSign, Inc. US

This certificate is valid from Mon Jan 29, 1996 to Wed Jan 07, 2004

This certificate is Trusted

4. Click the button to either Trust, or Untrust this particular certificate.
5. Reload the Listener.

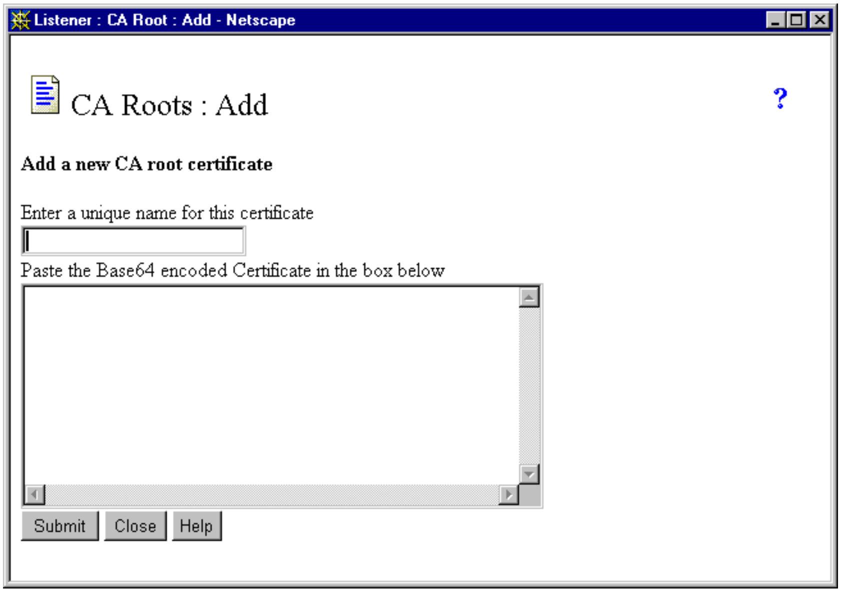
Adding a Certificate

To add a certificate:

1. Click Add.

   The CA Roots : Add form appears.

Figure 3–5 CA Roots: Add form



Add a new CA root certificate

Enter a unique name for this certificate

Paste the Base64 encoded Certificate in the box below

2. In the first box, enter a unique name for this certificate.

   (Limit is 512 bytes.)

3. Paste the Base64 Certificate in the second box.

Deleting a Certificate

To delete an existing certificate from the list, simply select the certificate and click the Delete button.

Upgrading an Existing Private Key File

This must be done if you are upgrading from a version prior to 4.0.8.

1. Enter the command **genreq**. The main menu appears:

```
Certificate Request Generator

G - Generate key pair and certificate request
U - Upgrade an old private key file (pre 4.0.8) to new for-
mat
```

```
Q - Quit
> Enter choice:
```

2. Enter U.

   ```
   u <RET>
   ```

3. Enter name of the old private key file.

   The default is **privkey.der.**

4. Enter name of the new private key file.

   The default is **`privkeyn.der.`**

5. From the main menu, enter “q” to quit.

4

Directory Servers

You can configure virtual paths such that users requesting access to the paths are verified against a directory server using X.509 digital certificates. Oracle Application Server can access the contents of directory servers that support LDAP v3 (lightweight directory access protocol).

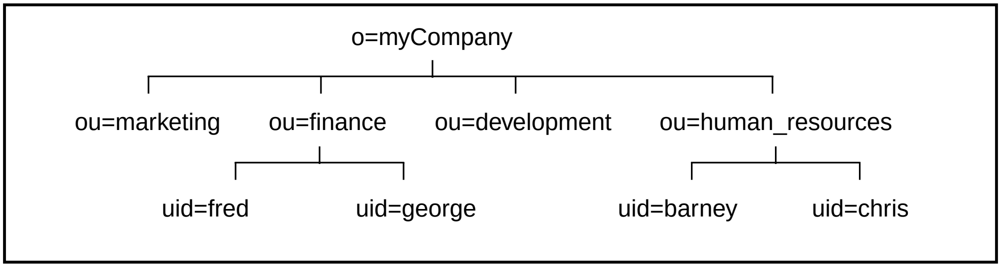
Oracle Application Server has been tested with LDAP-Compliant Server. See the directory server documentation for information on how to administer and set up users in the directory server.

Contents

Overview

Directory servers are used by organizations to define a hierarchical view of the organization’s employees, units, and other resources. For example, a directory server might contain the following entries:

Figure 4–1 Contents of a sample directory server



Directory servers use standard attribute names to define entries in the server. For example, "`o`" stands for organization, "`ou`" stands for organizational unit, and "`uid`" stands for user ID.

When you protect virtual paths using directory servers, you can limit access to the virtual paths to particular branches in a directory server. For example, given the structure above, you can specify that a virtual path can be accessed only by users in the finance branch.

Users identify themselves using X.509 v3 certificates. Oracle Application Server reads the certificate and checks if the user specified by the certificate is in the allowed portion of the directory server.

Note that you can use certificates and directory servers to protect only virtual paths associated with cartridges. You cannot use them to protect virtual paths associated with static files or CGI scripts.

Certificate Realms

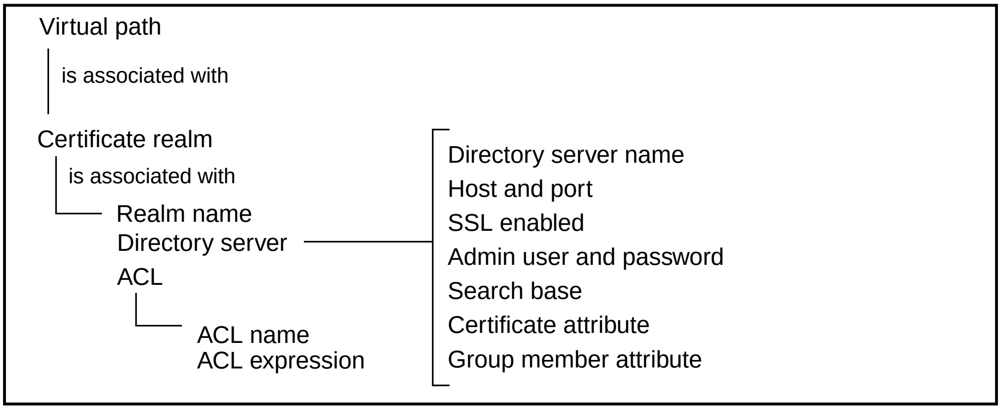
To protect virtual paths using directory servers, you associate these paths with certificate realms. A certificate realm consists of the following components:

Table 4–1 Components of a certificate realm

Component	Description
Realm name	Identifies the certificate realm.
LDAP server name	Identifies the directory server associated with this certificate realm.
ACL (access control list)	Specifies groups and users who have or do not have access.

The following figure shows the contents of each component:

Figure 4–2 Components of a certificate realm



When a user tries to access a virtual path that is protected by a certificate realm, the dispatcher calls the certificate provider. The certificate provider requires the following pieces of information to verify the user:

- The name of the certificate realm with which the virtual path is associated.
- The privilege type. For virtual paths, the privilege is “exec”.
- The certificate of the client.

When the user needs to send his or her certificate to the server, the user sees a dialog.

The certificate provider connects to the directory server associated with the realm and uses the ACL to locate the users who have access to the virtual path. It compares the client’s certificate with the certificates of allowed users.

Overview of Steps

To protect a virtual path using certificates and directory servers, you perform the following steps:

Task 1: Define Directory Servers

Task 2: Add the oraCertHash Attribute

Task 3: Run wraspcpp

Task 4: Define Access Control List (ACL)

Task 5: Define Certificate Realms

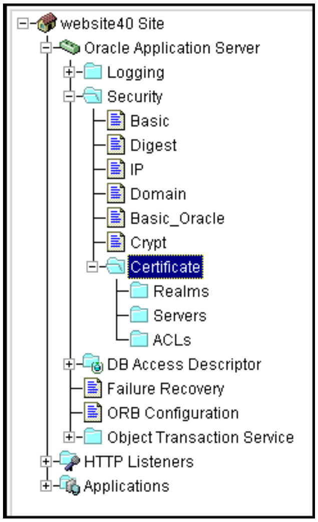
Task 6: Associate Virtual Path with Certificate Realm

Forms in the Oracle Application Server Manager

To configure directory servers, realms, and ACLs, use the Oracle Application Server Manager. To access these forms:

1. Start up your browser and connect to the Wecome page for Oracle Application Server.
2. Select OAS Manager.
3. Expand the name of the site you wish to configure.
4. Expand Oracle Application Server.
5. Expand Security.
6. Expand Certificate. You will see three options, Realms, Servers, and ACLs.

Figure 4–3 Certificate options



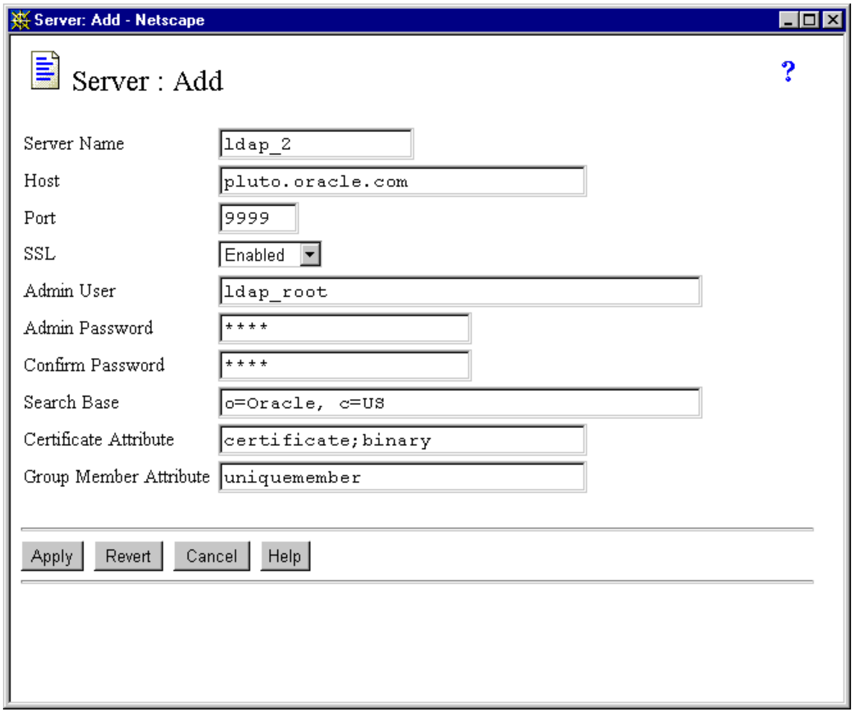
Task 1: Define Directory Servers

To define a directory server, use the OAS Manager.

1. Expand the name of the site you wish to configure.

2. Expand Oracle Application Server.
3. Expand Security.
4. Expand Certificate.
5. Click Servers.
6. Click ✚. This displays the Server:Add form.

Figure 4–4 Add Server form



Server : Add

Server Name	ldap_2
Host	pluto.oracle.com
Port	9999
SSL	Enabled
Admin User	ldap_root
Admin Password	****
Confirm Password	****
Search Base	o=Oracle, c=US
Certificate Attribute	certificate;binary
Group Member Attribute	uniquemember

7. In the Add Server form:
   - Server Name: enter a name to identify the directory server.
   - Host: enter a full name of the machine running the directory server.
   - Port: enter the port number at which the directory server is listening.
   - SSL: select whether or not SSL is used when Oracle Application Server communicates with the directory server.

- Admin User, Admin Password, and Confirm Password: enter the root DN and password to connect to the directory server.
- Search Base: enter the point (in DN format) in the directory server at which the search begins.
- Certificate Attribute: enter the name of the attribute in the directory server that contains the user’s binary certificate.
- Group Member Attribute: enter the name of the attribute in the directory server that specifies the group members.
- Click Apply.

8. Click OK in the Success dialog.

For the new directory server entry to appear in the navigational tree, hold down the Shift key while clicking the browser’s Reload button.

Directory Server Information in the Configuration File

Information about directory servers are stored in the [Auth.Cert.LDAPServer.<serverName>] section in the **wrb.app** configuration file. The following directives are valid in the section:

```
[Auth.Cert.LDAPServer.<serverName>]
Name = <ldap_server_name>
Host = <machine_running_ldap_server>
Port = <ldap_server_port>
SSL = yes | no
AdminUser = <admin_user_name>
AdminPwd = <admin_password>
SearchBase = <search_start_point>
CertificateAttr = <attribute_for_certificates>
GroupMemberAttr = <attribute_for_group_members>
```

The name of the directory server is also added to the LDAPServers directive in the [Cert] section in the **wrb.app** file.

The following example shows the configuration information for an LDAP server named ldap_2.

```
[Cert]
ACLs = finance
LDAPServers = ldap_2

[Auth.Cert.LDAPServer.ldap_2]
```

```
Name = ldap_2
Host = pluto.oracle.com
Port = 9999
SSL = yes
AdminUser = ldaproot
AdminPwd = LDKJFLSDKJFLDK
SearchBase = (o=oracle,c=US)
CertificateAttr = certificate;binary
GroupMemberAttr = uniquemember
```

Task 2: Add the oraCertHash Attribute

When you configure the directory server, you need to add the `oraCertHash` attribute to the list of user defined attributes. For example, for the Netscape directory server, you need to modify, either manually or through the administration interface, the **slapd.user_at.conf** file.

Task 3: Run wraspcpp

> **Note:** Please note that the **wraspcpp** utility and Oracle Application Server require a DER encoded certificate for LDAP-based authentication to work correctly.
>
> Certificates obtained from Internet Explorer or Netscape are in PKCS12 format, and are not suitable for direct loading into a directory.

Before Oracle Application Server can use a directory server, you need to run a utility (**wraspcpp**) to pre-calculate the certificates of users in the directory server. This utility sets the `oraCertHash` attribute for every user in the directory server who has a binary certificate.

Run the **wraspcpp** utility from the command-line, using the following syntax:

```
wraspcpp [-b yes|no] [-o yes|no]
```

`-b` (for batch) specifies whether **wraspcpp** runs in batch mode or in non-batch mode. In batch mode (`-b yes`), it does not prompt you. In non-batch or interactive mode (`-b no`), it prompts you. The default is batch mode.

`-o` (for output) specifies whether or not **wraspcpp** displays the entries in the directory server to which it added the `oraCertHash` attribute. The default is no.

wraspcpp works on all the directory servers defined in Oracle Application Server's configuration file. You must have defined at least one directory server in the configuration file before running **wraspcpp**.

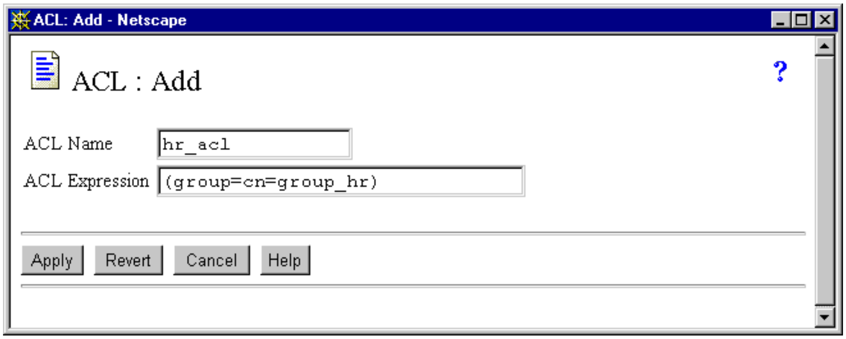
When you run **wraspcpp**, Oracle Application Server must be running. The reason for this is that **wraspcpp** uses the server to access the configuration data.

Task 4: Define Access Control List (ACL)

To define an ACL, use the Oracle Application Server Manager.

1. Expand the name of the site you wish to configure.
2. Expand Oracle Application Server.
3. Expand Security.
4. Expand Certificate.
5. Click ACLs.
6. Click ➕. This displays the ACL:Add form.

Figure 4–5 Add ACL dialog



ACL Name: hr_acl

ACL Expression: (group=cn=group_hr)

7. In the ACL:Add form:
   - ACL Name: enter a name to identify the ACL.
   - ACL Expression: enter the ACL. (See below for details.)
   - Click Apply.
8. Click OK in the Success dialog.

For the new ACL entry to appear in the navigational tree, hold down the Shift key while clicking the browser's Reload button.

ACL Format

An ACL consists of expressions with the following format:

```
(group=<string>)
(user=<string>)
```

where <*string*> specifies a DN. Use "group=" when the DN refers to a non-user; use "user=" when the DN refers to a user. Expressions are enclosed by parentheses.

You can combine ACL expressions using boolean operators: & (and), | (or), and ! (not).

ACL Examples

The following table shows some sample ACLs:

Table 4–2 Example ACLs

Example	Description
(group=ou=finance,o=myCompany,c=US)	Allow access to users who are in the finance group in myCompany in the US.
!(group=ou=finance,o=myCompany,c=US)	Deny access to users who are in the finance group in myCompany in the US.
(group=ou=finance,o=myCompany,c=US) | (group=ou=marketing,o=myCompany,c=US)	Allow access to users who are in the finance or marketing groups in myCompany in the US.
(user=uid=fred,ou=finance,o=myCompany,c=US)	Allow access to fred in the finance group in myCompany.

ACL Information in the Configuration File

ACLs for use with the certificate provider are stored in the [Auth.Cert.ACL.<acl_name>] section of the **wrb.app** configuration file. This section contains the following directives:

```
[Auth.Cert.ACL.<acl_name>]
Name = <acl_name>
Expr = <acl_expression>
```

The name of the ACL is also added to the ACLs directive in the [Cert] section in the **wrb.app** file, as shown in the following example:

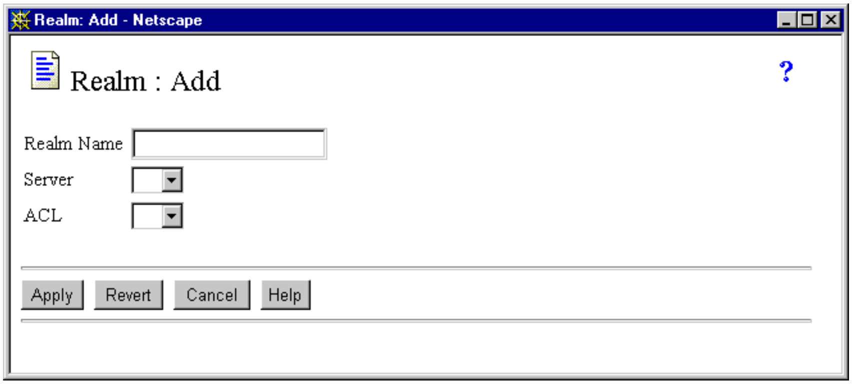
```
[Cert]
ACLs = finance
LDAPServers = ldap_2

[Auth.Cert.ACL.finance]
Name = finance
Expr = (group=ou=finance,o=myCompany,c=US)
```

Task 5: Define Certificate Realms

To define certificate realms, you use the Oracle Application Server Manager.

1. Expand the name of the site you wish to configure.
2. Expand Oracle Application Server.
3. Expand Security.
4. Expand Certificate.
5. Click Realms.
6. Click +. This displays the Add Realm form.

Figure 4–6 Add Realm form



7. In the Add Realm form:
   - Realm Name: enter a name to identify the realm.

- Server: select a directory server from the list. For information on how to define directory servers, see "Task 1: Define Directory Servers" on page 4-4.
- ACL: select an ACL from the list. For information on how to define ACLs, see "Task 4: Define Access Control List (ACL)" on page 4-8.
- Click Apply.

8. Click OK in the Success dialog.

For the new realm entry to appear in the navigational tree, hold down the Shift key while clicking the browser's Reload button.

Certificate Realm Information in the Configuration File

The certificate realm information is stored in the [Auth.Cert.Realm.<realm_name>] section of the **wrb.app** configuration file. This section contains the following directives:

```
[Auth.Cert.Realm.<realm_name>]
Name = <realm_name>
LDAPServer = <directory_server_name>
ACL.Exec = <acl_name>
```

The name of the realm is also added to the Realms directive in the [Cert] section in the **wrb.app** file, as shown in the following example:

```
[Cert]
ACLs = finance
LDAPServers = ldap_2

[Auth.Cert.Realm.test_realm]
Name = test_realm
LDAPServer = ldap_2
ACL.Exec = finance
```

Task 6: Associate Virtual Path with Certificate Realm

After you have defined certificate realms, you associate them with the virtual paths you want to protect. You do this using the Oracle Application Server Manager.

To protect virtual paths using certificate realms and directory servers:

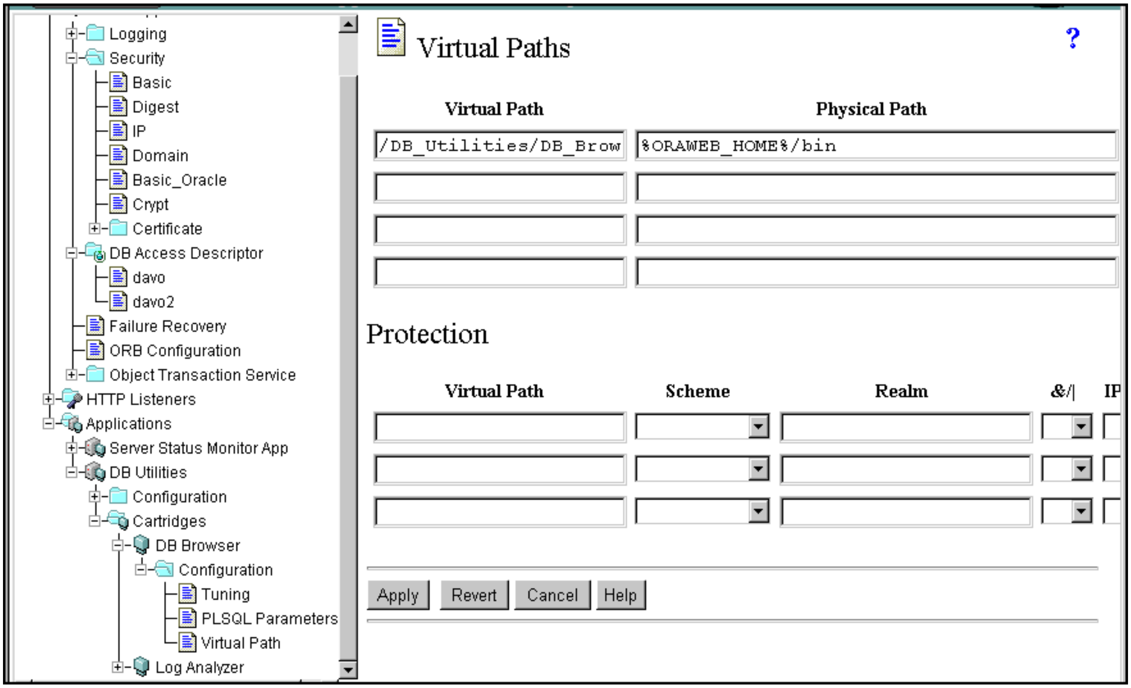
1. Start up your browser and connect to the Wecome page for Oracle Application Server.

2. Select OAS Manager.
3. Expand the name of the site you wish to configure.
4. Expand the Applications folder.
5. Expand the application you wish to configure.

   You will see two folders, Configuration and Cartridges.
6. Expand the Cartridges folder.
7. Expand the cartridge you want to configure.
8. Expand the Configuration folder.
9. Click on Virtual Path.

   This displays the Virtual Paths form.

Figure 4–7 Protecting a virtual path



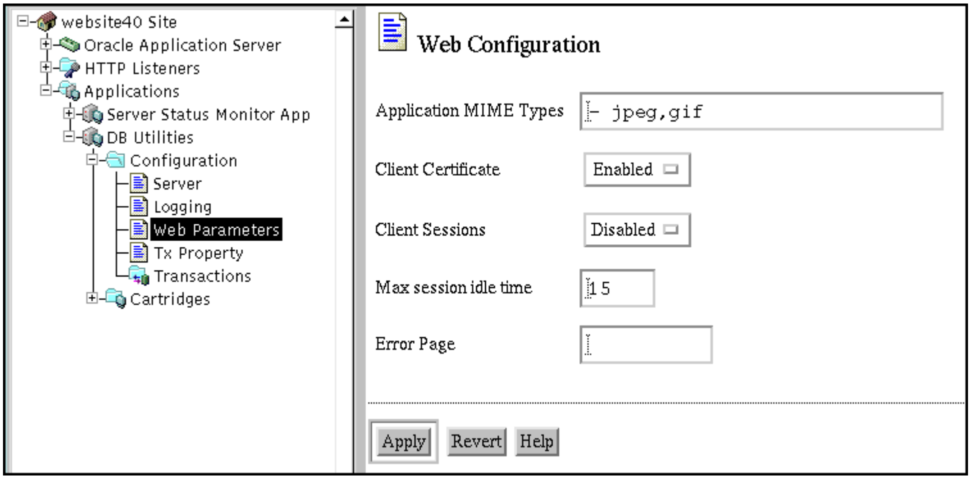
Virtual Path	Physical Path
/DB_Utilities/DB_Brow	%ORAWEB_HOME%/bin

10. In the Virtual Paths form:

- Virtual Path: enter the path you want to protect.
- Scheme: select Certificate.
- Realm: enter the name of a certificate realm to associate with the virtual path.
- &/|: select & if the user has to satisfy both the certificate and access control schemes. Select | if the user has to satisfy only one scheme.
- (optional) Domain/IP: select the access control scheme you want.
- (optional) Group: enter the name of an IP or domain group to associate with the virtual path. The group must exist in the selected scheme.

11. Click Apply.
12. At the same level as the Cartridges folder, expand the Configuration folder.
13. Click on Web Parameters to display the Web Configuration form.

Figure 4–8 Web Configuration form



Web Configuration

Application MIME Types: jpeg,gif

Client Certificate: Enabled

Client Sessions: Disabled

Max session idle time: 15

Error Page:

14. Set the Client Certificate parameter to Enabled.
15. Click Apply.
16. Restart Oracle Application Server for the new configuration to take effect.

wrb.app Information for Virtual Paths

When you associate a virtual path with a protection scheme, the following entries are added to the **wrb.app** file:

```
[AppProt]
<virtual_path> = <protect_string>
```

The following example shows a virtual path that is protected.

```
[AppProt]
/tr/owa/    = Cert(finance_realm)
```

5

Security for IIOP-based Applications: EJB, ECO/Java, and C++

Contents

Overview

When an external client submits a request for a CORBA object (EJB, ECO / Java, or C++ component), the request goes through the following security checks:

1. ORB Level Security Service: If the ORB has been configured to run in secure mode, every single IIOP request is subject to security checks. The ORB provides the following security features: authentication, access control, integrity, and encryption security features. If the client passes these security features, or if the client is part of the trusted environment, the ORB forwards the request to the application level.

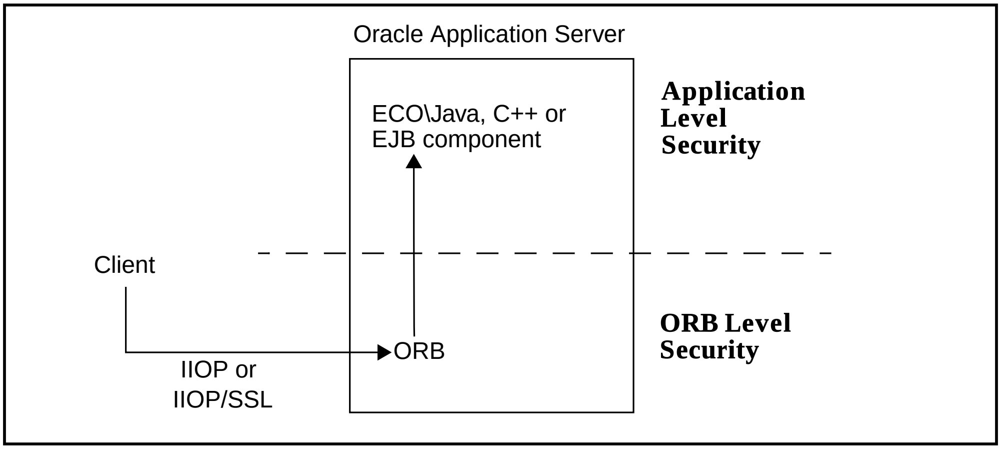
2. Application Level Security Service: The application level checks the request against the application level security defined. This includes:

   - authentication defined within the Auth Server
   - access control definitions contained within the Authentication String

If the request passes the security scheme definition or if it is from a client within the trusted environment, the request continues to be processed.

This is demonstrated in Figure 5–1.

Figure 5–1 Security options for IIOP applications


Oracle Application Server
ECO\Java, C++ or EJB component
Application Level Security
Client
ORB
IIOP or IIOP/SSL
ORB Level Security

The following table shows the different security features that these two security levels provide:

	ORB Level	Application Level
Authentication	Provides both host authentication (IP) and certificate based authentication (SSL certificate)	User authentication (username / password) and host authentication (IP and / or domain)
Access Control	Statically defined ACL	User configurable through Authentication String
Integrity	SSL	Provided by ORB level security service
Encryption	SSL	Provided by ORB level security service
Trust model	All components within the OAS site are trusted with each other	Only components within the same application are trusted with each other. This is valid for ECO / Java or EJB only.

The rest of this chapter discusses in detail how each level applies these security features.

ORB Level Security Service

When you set security features for the ORB, these options affect communication between all Oracle Application Server components, not just communication between external clients and CORBA applications. This means that the ORB verifies requests from all components before they are granted access.

Security Features

The ORB security service supports the following features:

Security Feature	ORB Level
Host Authentication	Provides both IP-based and certificate-based authentication (X.509 certificate used for SSL)
Access Control	Statically defined ACL
Integrity	SSL
Encryption	SSL
Trust model	All components within the OAS site are trusted

Host Authentication

There are two types of host authentication: IP-based and certificate-based authentication.

- IP-based Authentication—The host is authenticated by checking the client’s IP address. The client is granted access if the IP address is within the site.
- Certificate-based Authentication—Certificate-based authentication occurs by checking the X.509 certificate presented during SSL handshake. The client is granted access if the certificate presented is the site certificate.

Access Control

The application server has a single static ACL file that contains IP addresses or the site certificate of all the machines in the Oracle Application Server environment.

Integrity and Encryption

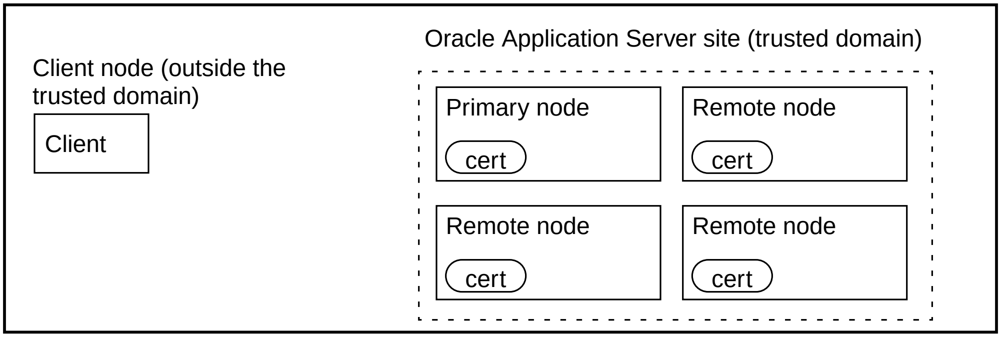
Components in Oracle Application Server are CORBA objects, and they use Internet Inter-ORB Protocol (IIOP) as the communication protocol. You can secure the IIOP channel from malicious snoopers by using IIOP/SSL, which provides integrity and encryption for all communications.

Trust Model

Any components that are co-located within the application server trust each other. They are considered to be in the “trusted domain”. They can access each other because the ACL contains the IP addresses of primary and remote nodes running Oracle Application Server, which applies to all components within the application server. If you enabled certificate verification, the trusted components still have access because all Oracle Application Server nodes contain the same certificate.

Clients running on machines that are not part of an Oracle Application Server site are considered to be outside the trusted domain.

Figure 5–2   Trusted and untrusted domains


Client node (outside the trusted domain)
Client
Oracle Application Server site (trusted domain)
Primary node
cert
Remote node
cert
Remote node
cert
Remote node
cert

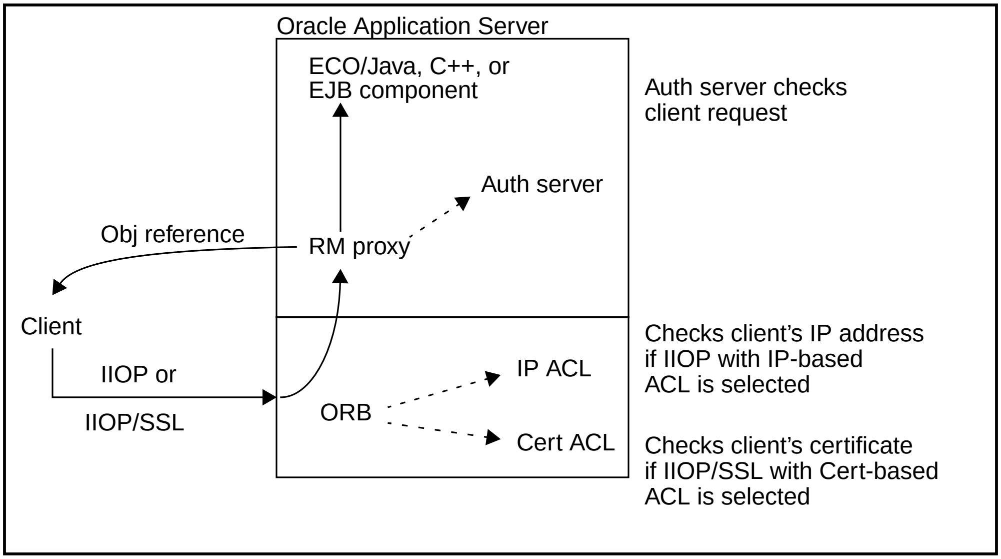
Trusted components include the applications you installed in the Oracle Application Server environment. This means that you should be aware of all the applications that you have running on your Oracle Application Server site since they have free access to Oracle Application Server components and also to any other applications in the Oracle Application Server environment. For example, if your application level security is not set (see “Application Level Security Service” on page 5-8), any JServlet application can access any ECO/Java and EJB applications without going through a security check. In addition, the application level security has its own trust model definition.

Security Options Available

You can choose one of the following options, which are combinations of security features discussed in "Security Features" on page 5-3, to set security for the ORB:

- IIOP with IP-based Access Control List (ACL)—This defines access control through IP addresses only. All communication is over IIOP.
- IIOP over SSL with Certificate-based ACL—This defines authentication integrity and encryption through IIOP / SSL certificate-based access control.

Figure 5–3 Security options


Oracle Application Server
ECO/Java, C++, or EJB component
Auth server checks client request
Auth server
Obj reference
RM proxy
Client
IIOP or
IIOP/SSL
ORB
IP ACL
Cert ACL
Checks client's IP address if IIOP with IP-based ACL is selected
Checks client's certificate if IIOP/SSL with Cert-based ACL is selected

IIOP with IP-based Access Control List (ACL)

You can enable the ORB to check the client's IP address. The ORB grants access to the client only if the client's IP address is authorized by the ACL used by the ORB.

> **Note:** When you select this option, Oracle Application Server uses plain IIOP (instead of IIOP / SSL) as the communication protocol.

The ORB authenticates clients using the client's IP address. If the IP address is listed in the ACL under the list of allowed addresses, then the client is granted access. Otherwise, the client is denied access. You cannot modify the IP-based ACL. The ACL contains IP addresses of all the machines in the Oracle Application Server environment.

To use this option:

- Set the security option to "IIOP + IP-based ACL" in the ORB General Parameters form. See "Configuring ORB-Level Security Service" on page 5-7 for details.

> **Note:** This option is not completely secure as intruders can spoof IP addresses.

When this option is in effect, the only clients that can access CORBA applications are those running on machines that are primary or remote nodes of Oracle Application Server. These are the only machines whose IP addresses are listed in the ACL.

IIOP over SSL with Certificate-based ACL

Checking certificates is more secure than checking IP addresses. When you enable the certificate option, the communication between a client and Oracle Application Server is IIOP / SSL, not just plain IIOP. This provides for authentication, integrity, and encryption.

When you configure the ORB to run in IIOP / SSL mode, the ORB automatically requests the client's X.509 certificate in every single connection. The ORB grants access to the client only if the client's certificate is equal to the site certificate in the ACL used by the ORB. The site certificate is the certificate used by OAS for all nodes within an OAS site. If ORB is SSL-enabled, the Distinguished Name (DN) of the site certificate will be included as part of ORB ACL. Every request will be checked against this DN. If the client certificate of a particular request does not match the DN of this site certificate, the request is rejected.

This option provides authentication by requesting the client's X.509 certificate. This certificate must be the site certificate in order to access protected methods.

To use this option:

- Get an X.509 certificate from a CA.
- Install the certificate for Oracle Application Server. To do this, you install a wallet using the Oracle Wallet Manager on the primary node. A wallet contains the certificate and trustpoints. See Chapter 6, "Oracle Wallet Manager" for details.

- Set the security option to “IIOP / SSL + Cert-based ACL” in the ORB General Parameters form. See “Configuring ORB-Level Security Service” on page 5-7 for details.

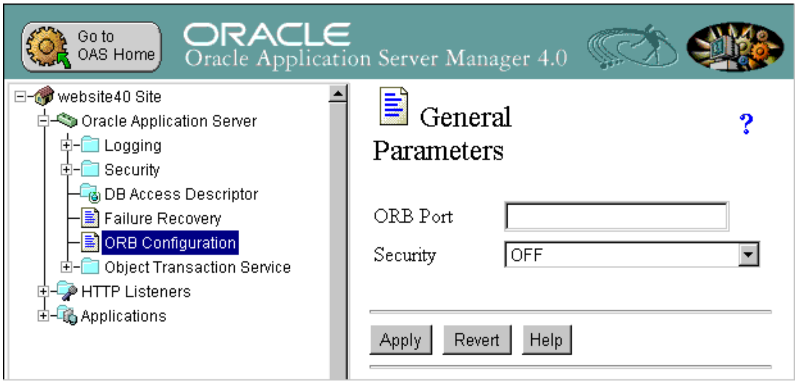
When this option is in effect, CORBA clients can access protected components in Oracle Application Server only when they present the Oracle Application Server site certificate.

Configuring ORB-Level Security Service

Use the ORB General Parameters form to set the security option.

1. Start up your browser and go to the Oracle Application Server Welcome page.
2. Select OAS Manager.
3. Expand the name of the site you wish to configure.
4. Expand Oracle Application Server.
5. Click ORB Configuration to display the General Parameters form.

Figure 5–4 ORB General Parameters form



ORB Port

Security OFF

6. In the ORB General Parameters form:
   - ORB Port: Not related to security and should already be set. The ORB Port is the TCP port number where the ORB listens for incoming connection requests.
   - Security: choose a security option:
     - * OFF - to disable security. This is the default.

  - IIOP + IP-based ACL
  - IIOP / SSL + Cert-based ACL

7. Click Apply.

When you click Apply, the Oracle Application Server Manager stops the ORB processes and updates the ACL in the ORB accordingly:

- If you enabled IIOP + IP-based ACL, it updates the ACL to include the IP addresses of all hosts in an Oracle Application Server site, and propagates the updated ACL to the remote nodes.
- If you enabled IIOP / SSL + Cert-based ACL, it reads the certificate that you installed on the primary node, updates the certificate in the ACL, and propagates the updated ACL to the remote nodes.

Note: When you change any of the options on the General Parameters form, Oracle Application Server is stopped and restarted for the new options to take effect.

Application Level Security Service

The following list and describe the security features available for your IIOP-based application:

Security Feature	Application Level
Authentication	User authentication (username / password) and host authentication (IP and / or domain). For an external client to access a protected EJB, ECO / Java or C++ application, you can require it to provide user authentication through a username / password or host authentication by coming from an acceptable IP address or domain.

Security Feature	Application Level
Access Control	User configurable through authentication string. See Chapter 2, “Authentication Schemes” for a definition of authentication string. For an external client to access a protected EJB, ECO / Java or C++ application, it has to pass the security restrictions imposed as defined within the authentication string. ■ EJB and ECO / Java—The authentication string applies to all components within the application. ■ C++—The authentication string applies to all components within the cartridge.
Integrity	Provided by ORB level security service
Encryption	Provided by ORB level security service
Trust model	Only clients within the same application are trusted. Any EJB or ECO / Java component within an application can access another component within the same application without needed to provide the required security.

Authentication

There are defined security schemes that apply to all application types. These are defined in Chapter 2, “Authentication Schemes”. The application level security service uses the security schemes to define its authentication policy.

- User Authentication is provided through the Basic or Basic_Oracle security schemes.
- Host Authentication is provided through the IP or Domain security schemes.

For EJB, ECO / Java or C++ applications, the supported security schemes are listed in Table 5–1.

Table 5–1 Security schemes for EJB, ECO/Java or C++ applications

Scheme	Description
Basic	You define username/password pairs, which are stored in the application server. The password is stored in an encrypted format. To access an application protected by the Basic scheme, a client needs to provide a valid username/password.
Basic_Oracle	To access an application protected by the Basic_Oracle scheme, a client needs to provide username/password that can be used to log into an Oracle database. If the login fails, the client is denied access. This scheme is convenient for cases where valid users already have database accounts as you do not have to duplicate username/password information in the application server.
IP	You define a list of IP addresses that can or cannot access the application. The addresses are stored in the application server's configuration file. To access an application protected by the IP scheme, a client must have an IP address that can access the application. See the note on IP and Domain schemes below.
Domain	Similar to the IP scheme, except that the domain scheme uses domain names instead of IP addresses. See the note on IP and domain schemes below.
noaccess	In this scheme, an EJB application can only be accessed by other EJB applications/objects in the application server or other application server components such as Java Servlet applications. It cannot be accessed by external clients directly. This scheme is only available for IIOP-based applications.

Once you define the security scheme that you desire, you must configure both the client and the server to require the combination of security schemes desired for the application. This configuration occurs within an authentication string. See "Access Control: Defining the Authentication String" on page 5-11 for more information on defining and using authentication strings.

Access Control: Defining the Authentication String

To protect EJB applications from unauthorized external clients, you associate them with authentication strings that define the type of protection. Even though the term "authentication string" is used, it provides the definition for access control.

The same schemes must be set up on both the EJB application and the client:

EJB, ECO / Java or C++ application

An authentication string for the desired security level is created and set within the application deployment descriptor or deployment file.

However, for EJB or ECO / Java components, a component can access another component within the same application without being authenticated. See “Trust Model” on page 5-17 for more information.

External client

Security level is declared within JNDI environment properties (programmatic) or inherited through the client’s environment (declarative). See “Enabling Authentication for Clients” on page 5-18 for more information.

You need to set an authentication string format within the server. This string is also used by the client to access the server. Once you have decided how to set up your authentication string, configure the authentication string within the application either through the deployment descriptor (EJB applications) or deployment file (ECO / Java applications). The following sections explain how to specify authentication strings:

- Specifying Authentication Strings within EJB
- Specifying Authentication Strings for ECO / Java Applications
- Specifying Authentication Strings for C++ Applications

Note: Enterprise JavaBeans security features are in a state of change. The 1.0 security features have been deprecated. The 1.1 specification will be supported in a future release.

The digest and certificate authentication scheme is not supported.

For IP and Domain schemes, if the client comes through an IIOP proxy, it is likely that the client's IP address is that of the proxy server sitting at the firewall, not the true client's IP address. In this case, the application server grants access to the client only if the authentication information set up for the EJB application includes the proxy server's IP address.

Authentication String Format

To protect an application with an authentication server scheme, you assign an authentication string to the application. The authentication string has the following format:

```
<scheme>(<realm>) [ {"|" | "&"} & <scheme>(<realm>) ... ]
```

<*scheme*>	*scheme* specifies an authentication server scheme. It is one of: `Basic`, `Basic_Oracle`, `IP`, `Domain`, or `noaccess`. If the scheme is `noaccess`, the realm is empty, and the authentication string looks like: `noaccess()`
<*realm*>	*realm* specifies a realm in the specified scheme. See Chapter 2, "Authentication Schemes" for information on how to define realms.
&	An authentication string can consist of more than one "scheme(realm)" specification. For example, you can have an authentication string that looks like: `Basic(myProject) & IP(buildingOne)` This authentication string consists of two parts, and a client must fulfill both parts to access the protected EJB applications. The client must provide a username/password for the "myProject" realm and its IP address must be in the "buildingOne" realm.

<table>
<tr><td>|</td><td>You can also connect the parts with the | character. This “or” operator indicates that a client needs to fulfill only one part of the authentication string. For example, in the following authentication string:
Basic(myProject) | IP(buildingOne)
a client can access the application if it provides a username/ password for the “myProject” realm, or if its IP address is in the “buildingOne” realm.</td></tr>
</table>

Note: There are no precedence order rules for the "|" and "&" symbols in the authentication scheme. In addition, you cannot force a precedence with parenthesis.

Specifying Authentication Strings within EJB

You can specify authentication strings for EJB applications in either the application-level deployment descriptor or ECO.APP. See “Specifying Authentication Strings for ECO/Java Applications” on page 5-14 for information on configuring your authentication string within the ECO.APP file.

To specify your authentication string programmatically in the deployment descriptor, use the method `setAuthenticationString()` in the class `oracle.oas.ejb.deployment.OASApplicationDescriptor`. The method’s sole parameter is an authentication string. If `setAuthenticationString()` is not invoked, the EJB application is unprotected. Refer to "Creating Deployment Files" in *Oracle Application Server Developer’s Guide: EJB, ECO/Java and CORBA Applications* for deployment descriptor details.

Use the `noaccess` scheme to define EJB applications that can be accessed only from within the application server.

The following code segment defines the deployment descriptor for an EJB application protected with the Basic_Oracle scheme:

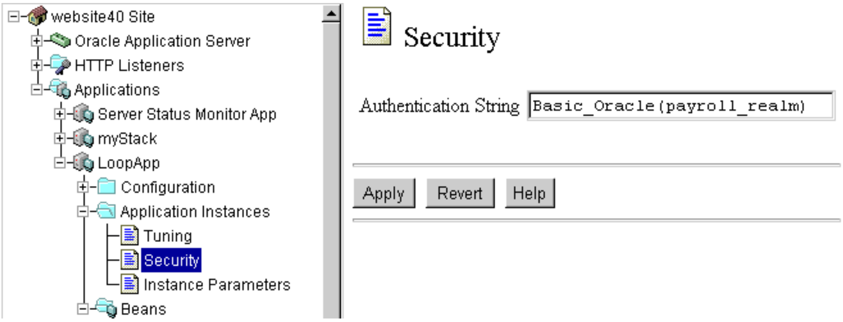
```
OASApplicationDescriptor ad = new OASApplicationDescriptor();
ad.setSessionTimeout(2000);                  // set the timeout to 2000 seconds

// set the authentication string for the application
ad.setAuthenticationString("Basic_Oracle(payroll_realm)");

// set the ControlDescriptors for the EJB application
// cdArray is an array of ControlDescriptors, one cell for each bean. For
// example, if there is only one bean in this application, we only need a
// one-cell array
ControlDescriptor cd = new ControlDescriptor();
ControlDescriptor cdArray[] = new ControlDescriptor[1];

cdArray[0] = cd;
ad.setControlDescriptors(cdArray);
```

The following example shows deployment descriptor for an object protected with the Basic and IP schemes:

```
OASApplicationDescriptor ad = new OASApplicationDescriptor();
ad.setSessionTimeout(2000);                  // set the timeout to 2000 seconds

// set the authentication string for the object
ad.setAuthenticationString("Basic(hr) & IP(hq)");

// set the ControlDescriptors for the EJB application
// cdArray is an array of ControlDescriptors, one cell for each bean. For
// example, if there is only one bean in this application, we only
// need a one-cell array
ControlDescriptor cd = new ControlDescriptor();
ControlDescriptor cdArray[] = new ControlDescriptor[1];
```

You can change the authentication string for EJB applications after you install them in the application server. Use the Security form to do this.

Figure 5–5 Security form for EJB applications



Authentication String: Basic_Oracle(payroll_realm)

Specifying Authentication Strings for ECO/Java Applications

You specify authentication strings for ECO/Java objects in the **ECO.APP** file. The parameter name is `authenticationString`, and its value is an authentication string. The parameter appears in the `[<object>]` section of **ECO.APP**. For a full description on creating ECO.APP files, see *Oracle Application Server Developer's Guide: EJB, ECO/Java and CORBA Applications*.

If an object does not have the `authenticationString` parameter, it is an unprotected object.

Authentication strings apply only to clients that are external to the application server. If a client is another ECO/Java object, application, or other application server components—such as a Java Servlet application—the access restrictions do not apply. Use the `noaccess` scheme to define ECO/Java applications that can be accessed only from within the application server.

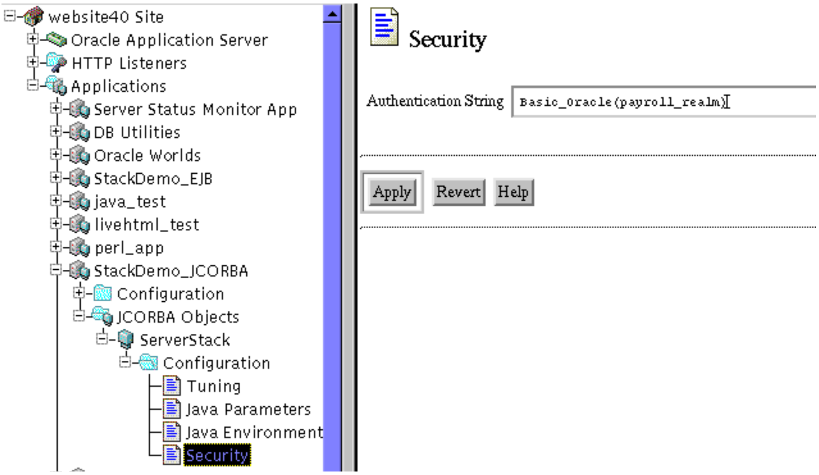
The following sample **ECO.APP** contains two protected objects:

```
[APPLICATION]
name = Payroll
idleTimeOut = 2000
transactions = ENABLED
transactionalDads = payrolldad

[employees]
className = emp
remoteInterface = emp_interface
transactionMode = TX_REQUIRED
authenticationString = Basic_Oracle(payroll_realm)
```

```
[benefits]
className = benefit
remoteInterface = benefits_interface
transactionMode = TX_REQUIRED
authenticationString = Basic(hr) & IP(hq)
```

You can change the authentication string for ECO / Java objects after you install them in the application server. You use the Security form to do this.

Figure 5–6 Security form for ECO/Java applications



Authentication String: Basic_oracle(payroll_realm)

Specifying Authentication Strings for C++ Applications

You can specify an authentication string for a C++ application in the deployment descriptor file **CPP.app**. Following is an example of the application section of **CPP.app** for a bank application:

```
[APPLICATION]
name = bank
timeout = 6000
transactions = Enabled
transactionalDads = BANK
authenticationString = Basic

[Account]
remoteInterface = Bank::Account
implementationClass = Bank::AccountImpl
implementationHeader = AccountImpl.h
```

```
timeout = 1000
stateless = false
authenticationString = Basic
```

In [APPLICATION] the section, the line `authenticationString = Basic` describes that the application bank is protected by the basic security scheme. The `authenticationString` property is optional: by default, an application does not require any authentication.

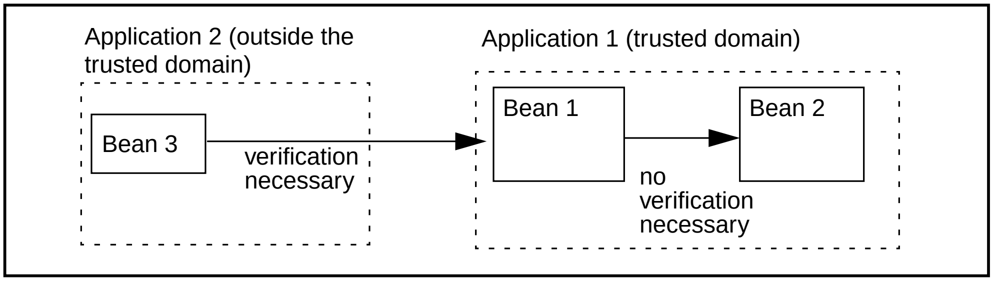
You can also set the `authenticationString` property for each cartridge in a C++ application. If this property is not specified, it will take its value from the application authentication string. If cartridge authentication string is specified, it will take precedence over the application authentication string.

See Chapter 4 "Creating the Deployment Descriptor File" of the *Oracle Application Server Developer's Guide: C++ CORBA Applications*.

Trust Model

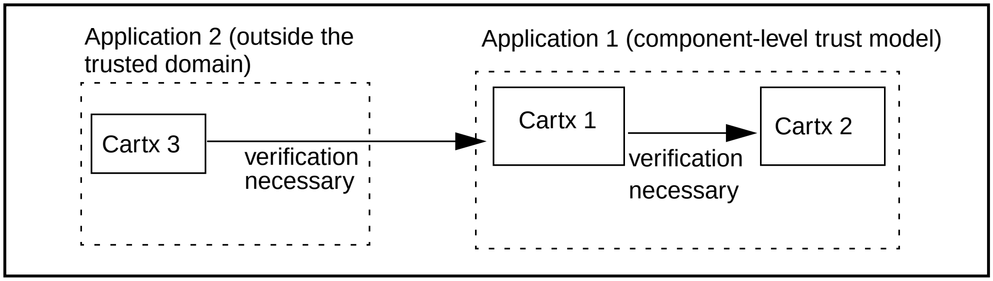
For EJB and ECO / Java applications, components within the same application do not need to verify themselves against any of the security features. They are automatically allowed access to another component within the same application. See Figure 5–7.

Figure 5–7 EJB and ECO/Java trusted and untrusted domains


Application 2 (outside the trusted domain)
Application 1 (trusted domain)
Bean 3
verification necessary
Bean 1
Bean 2
no verification necessary

For C++ applications, every component within an application has its own trust domain. Therefore, all components within an application need to verify themselves against any of the security features. As described in Specifying Authentication Strings for C++ Applications section, C++ applications can have component-level (cartridge-level) security as well as application level security. Even if the security of a cartridge is not specified, it needs to verify itself to access another cartridge within the same application. See Figure 5–8.

Figure 5–8 C++ trusted and untrusted domains


Application 2 (outside the trusted domain)
Cartx 3
verification necessary
Application 1 (component-level trust model)
Cartx 1
Cartx 2
verification necessary

Enabling Authentication for Clients

Before a client can access a protected application, it must provide the necessary information for authentication based on how the application is protected. The client needs to set its authentication string to declare the same security scheme as the server application.

The following table demonstrates what the client is required to provide based upon the security scheme used by the application:

Table 5–2 Client requirements for each Application Security Scheme

Basic and Basic_Oracle	Client sets SECURITY_PROTOCOL to indicate the scheme that the application uses. In addition, the client must provide a username and password within SECURITY_PRINCIPAL and SECURITY_CREDENTIALS.	Set SECURITY_PROTOCOL to one of the following values: AM_ORACLE_DEFAULT AM_SHAREDPASSWD AM_SHAREDPASSWD_MD_5 AM_SHAREDPASSWD_DB
IP and Domain	Client sets SECURITY_PROTOCOL to indicate which of the schemes that the application uses. Other required information is passed along within the client's IIOP request.	Set SECURITY_PROTOCOL to one of the following values: AM_IP AM_DOMAIN
noaccess	No client settings required.	

The following table lists the possible values for the SECURITY_PROTOCOL property. How to set SECURITY_PROTOCOLS are explained later in this chapter.

Table 5–3 Values for the SECURITY_PROTOCOL property

Value	Description
AM_ORACLE_DEFAULT	The values in SECURITY_PRINCIPAL and SECURITY_CREDENTIALS are authenticated appropriately against the protection schemes specified in the application's authentication string. The username / password information is passed in the clear. For example, if an application is protected with the Basic scheme, the username / password information is authenticated against the values in the server's configuration file. If an application is protected with the Basic_Oracle scheme, the username / password information is used to log into an Oracle database.
AM_SHAREDPASSWD	The values in SECURITY_PRINCIPAL and SECURITY_CREDENTIALS are passed in the clear.
AM_SHAREDPASSWD_MD_5	The value in SECURITY_CREDENTIALS is passed in MD5 format.
AM_SHAREDPASSWD_DB	The values in SECURITY_PRINCIPAL and SECURITY_CREDENTIALS are passed in the clear, and the authentication provider tries to log into an Oracle database using those values.
AM_IP	The values in SECURITY_PRINCIPAL and SECURITY_CREDENTIALS are not used because the application server determines the client's IP address from the request.
AM_DOMAIN	The values in SECURITY_PRINCIPAL and SECURITY_CREDENTIALS are not used because the application server determines the client's domain name from the request.

The information provided by the client is authenticated by the Auth Server, which is then checked against the application's authentication string, and if the verification passes, the server allows the client to access the application. Table 5–4 shows how clients are authenticated when they try to access secure EJB, ECO / Java or C++ applications.

Note: In the table, User Authentication scheme is used to refer to Basic scheme or Basic_Oracle scheme while Host Authentication scheme is used to refer to IP scheme or Domain scheme.

To use this table, determine the authentication scheme for the application in the first column, and check the row to determine the information a client needs to provide to get access. For example, if an application is protected by “User Auth scheme & Host Auth scheme”, a client can access the application only if it uses AM_ORACLE_DEFAULT or AM_SHARED_PASSWD* methods as the protocol, provides a valid username/password, and is running on a host with a valid IP or domain name.

Table 5–4 Effects of providing various values for authentication

Client JNDI Environ. Props / Server Schemes	AM_ORACLE_DEFAULT without username/ password	AM_ORACLE_DEFAULT with username/ password	AM_SHAREDPASSWD* with username/ password	AM_IP or AM_DOMAIN
User authentication scheme (Basic or Basic_Oracle)	Exception thrown.	Verify username/password against all realms listed in the authentication string.	Verify username/password against realms listed in the authentication string	Exception thrown.
Host authentication scheme (IP or Domain)	Authenticate the client’s address.	Authenticate the client’s address.	Exception thrown.	Authenticate the client’s address
User Auth scheme | Host Auth scheme	Authenticate against restriction scheme.	Authenticate against both schemes. If one scheme is valid, the client is granted access.	Authenticate against both schemes. If one scheme is valid, the client is granted access.	Authenticate against both schemes. If one scheme is valid, the client is granted access.
User Auth scheme & Host Auth scheme	Exception thrown.	Authenticate against both schemes. Client is granted access only if both schemes are valid.	Authenticate against both schemes. Client is granted access only if both schemes are valid.	Exception thrown.

* AM_SHAREDPASSWD includes the AM_SHAREDPASSWD_MD_5 and the AM_SHAREDPASSWD_DB protocols.

The following sections describe how to set security scheme values for each type of client:

- Setting Authentication for EJB or ECO/Java Clients
- Setting Authentication for C++ Clients

Setting Authentication for EJB or ECO/Java Clients

An EJB or ECO / Java client provides the authentication information using JNDI environment properties. The details for the JNDI environment properties which need to be set by a client are shown in the table below.

Table 5–5 Security environment properties set by the application

Property	Description
javax.naming.Context. SECURITY_PROTOCOL	This is a String value that indicates which security scheme the application is expecting. In addition, this field indicates if a username and password are provided within the SECURITY_PRINCIPAL and SECURITY_CREDENTIALS properties. Valid values are listed in Table 5–3.
javax.naming.Context.SECURITY_PRINCIPAL	This is a String value that specifies the username to be authenticated.
javax.naming.Context.SECURITY_CREDENTIALS	This is either a String value or byte[] array value that identifies the password or scheme-specific authentication data. If the value is in String format and SECURITY_PROTOCOL is set to AM_SHAREDPASSWD_MD_5, the application server assumes that the value is in clear text and converts it to MD5 format before sending it to the authentication server.

The EJB or ECO / Java client can set these properties either programmatically or declaratively:

- Setting the Client's Security Programmatically
- Setting the Client's Security Declaratively

Setting the Client's Security Programmatically

When a client's code specifically sets the JNDI environment properties with authentication information, security is being enabled programmatically.

The following example shows an application client that sets the security features using data that it gets from calling its own user-defined methods.

```
public class app_client {
  public static void main( String args[]) {
    myApp.TextFuncsRemote tfRemote;
    String user, password;
    TextFuncsRemote tfRemote = null;

    //create an environment object for JNDI properties
    // Note that the package is not oracle.oas.jndi, but is
    // oracle.oas.naming.jndi. If you use oracle.oas.jndi,
    // you will encounter problems.
    Hashtable env = new Hashtable();
    env.put(javax.naming.Context.URL_PKG_PREFIXES, "oracle.oas.naming.jndi");

    user = get_username();    // user-defined methods to get the user"s username
    password = get_password();// and password

    // set the JNDI security properties to support Basic or Basic_Oracle scheme
    env.put(javax.naming.Context.SECURITY_PROTOCOL, "AM_ORACLE_DEFAULT");
    env.put(javax.naming.Context.SECURITY_PRINCIPAL, user);
    env.put(javax.naming.Context.SECURITY_CREDENTIALS, password);

    // create the JNDI initial context with desired security.
    javax.naming.Context initialContext = new InitialContext(env);

    try {

     // retrieve the application through the initial context
     TextFuncsHome tfHome = (TextFuncsHome) PortableRemoteObject.narrow(
            initialContext.lookup("oas://host:8000/myApp/TextFuncsHome"),
                                  TextFuncsHome.class);

     tfRemote = tfHome.create();

     // If user/password is not valid, the javax.naming.NamingException
     // exception is thrown.

    } catch (javax.naming.AuthenticationException e) { // verify code
      e.getRootCause().printStackTrace();
      return;
    }

    // invoke methods on the object, if authentication succeeded

    tfRemote.remove(); // destroy object when done
  }
}
```

Setting the Client's Security Declaratively

Instead of setting security parameters programmatically, a client may set the security through either the command line or the Applet parameters. If the client sets the security parameters declaratively, it must use the following:

To set this ...	Use this ...
javax.naming.Context.SECURITY_PROTOCOL	java.naming.security.protocol
javax.naming.Context.SECURITY_PRINCIPAL	java.naming.security.principal
javax.naming.Context.SECURITY_CREDENTIALS	java.naming.security.credentials

- Java applications: security information can be passed on the execution line. The following example shows the client, myClient, invoked with the security set to AM_ORACLE_DEFAULT with a username and password.

```
java  -Djava.naming.security.protocol=AM_ORACLE_DEFAULT
      -Djava.naming.security.principal=usrname
      -Djava.naming.security.credentials=passwd
      myClient
```

- Java applets: security information can be passed in the <PARAM> tags for the APPLET. The following example shows the applet configured with the security set to AM_ORACLE_DEFAULT with a username and password.

```
<APPLET CODE=AppletSubclass.class WIDTH=anInt HEIGHT=anInt>
<PARAM NAME=java.naming.security.protocol VALUE=AM_ORACLE_DEFAULT>
<PARAM NAME=java.naming.security.principal VALUE=usrname>
<PARAM NAME=java.naming.security.credentials VALUE=passwd>
</APPLET>
```

Setting Authentication for C++ Clients

C++ clients of a protected C++ cartridge cannot use the CORBA `CosNaming::NamingContext` interface, since the CORBA CosNaming interface does not provide explicit support for passing security parameters. Oracle Application Server has expanded the CosNaming interface into a new interface called **SecNamingContext** (`oracle::oas::naming::SecNamingContext`).

Using SecNamingContext

SecNamingContext requires clients to use `secure_resolve`() for protected C++ cartridges. The `secure_resolve`() method requires the client to give security parameters like username, password, etc. corresponding to the authentication scheme used to protect the C++ cartridge. The secure_resolve() method returns the object reference of the C++ object as a name:

```
interface SecNamingContext:CosNaming::NamingContext
{
    //
    :

    Object secure_resolve(  in ::CosNaming::Name n,
        in AuthenticationMethod method,
        in string security_name,
        in Opaque auth_data,
        out Opaque auth_specific_data)
        raises(::CosNaming::NamingContext::NotFound,
        :CosNaming::NamingContext::CannotProceed,
        :CosNaming::NamingContext::InvalidName);

};
```

Using Message Digest

You can use message digests if your C++ client want to avoid sending passwords as cleartext. Message digest functions (in conjunction with other encryption technologies) are widely used to provide digital fingerprints for files.

When a C++ cartridge is protected by the basic authentication scheme, the clients of this cartridge can access it only by providing valid usernames and passwords. Since passwords are security sensitive data we allow clients to send MD5 digests of passwords instead.

The class oas::cpp::security::MessageDigest encapsulates an engine that generates message digests from inputs of arbitrary length. Message digest functions are one-way functions that exhibit the following properties:

- Given a hash value, it is computationally infeasible to find the original input that hashes to the value.
- On average, a single bit change of the input affects 50% of the output bits.

Note that simply choosing to send MD5 hashes of passwords do not prevent replay attacks. We recommend the use of IIOP/SSL to provide confidentiality and integrity of IIOP traffic. Since this is an abstract class, you need to create instances of this class by using the oas::cpp::MessageDigestFactory class.

Example

The following code snippet shows how you can use `secure_resolve()` in your client code to access a protected C++ cartridge.

```
int main (int argc, char **argv)
{

    CORBA::ORB_var the_orb = CORBA::ORB_init(argc, argv);
    oracle::oas::naming::SecNamingContext_var inc = NULL;

    // Obtain the IOR for the initial naming context using the
    // C++ cartridge Name Service boot strap mechanism.
    try {
        const char*             obj_ref = NULL;
        CORBA::Object_ptr       obj = NULL;

        obj_ref = oas::cpp::NSBootStrap::getOASRootNamingContext(argv[1]);
        obj =  CORBA::ORB::string_to_object(obj_ref);
        oas::cpp::NSBootStrap::freeIOR(obj_ref);
        inc = oracle::oas::naming::SecNamingContext::_narrow(obj);

    } catch (...) {
        cerr << "Unknown error obtaining initial naming context" << endl;
        exit(1);
    }

    try {

        // Creates a CosNaming name object to access the c++ cartridge
        CosNaming::Name cartx_name;
        name.length(2);
        name[0].id = CORBA::string_dup("CppApp_XXX");
        name[1].id = CORBA::string_dup("CppCartx_YYY");


        // Selects the proper authentication method
        oracle::oas::naming::AuthenticationMethod auth_method =
        oracle::oas::naming::SecNamingContext::AM_SHAREDPASSWD;

        // Initializes Authentication Data
        const char * user_name = "scott";
        const char * passwd = "tiger";

        oracle::oas::naming::Opaque auth_data(1024);
        // MD5 special case
        if ( auth_method ==
        oracle::oas::naming::SecNamingContext::AM_SHAREDPASSWD_MD_5 ) {

            oas::cpp::security::MessageDigest *md =
            oas::cpp::security::MessageDigestFactory::getInstance("MD5");
            md->update((unsigned char *)passwd,0,strlen(passwd));
            unsigned char *out_buf = new unsigned char[16];
            md->digest(out_buf);

            auth_data.length(16);
            for (int i=0; i<cred_len; i++) {
            auth_data[i] = (CORBA::Octet)out_buf[i];
            }

        } else { // Other cases

            int len = 0;
            if (passwd != NULL)
            len = strlen(passwd);

            auth_data.length(len);
            for (int i=0; i<len; i++) {
            auth_data[i] = (CORBA::Octet)passwd[i];
        }

        }

        oracle::oas::naming::Opaque_var token;
        oracle::oas::naming::Opaque_out auth_specific_data(token);

        // Calls secure_resolve
        CORBA::Object_ptr cpp =
        snc->secure_resolve(cartx_name,
                                  auth_method,
                                  user_name,
                                  auth_data,
                                  auth_specific_data);

    } catch (...) {
        cerr << "Unknown error" << endl;
        return 1;
    }
    return 0;
}
```

See Chapter 7 "Reference" of the *Oracle Application Server Developer's Guide: C++ CORBA Applications* for more on Oracle Application Server authentication schemes API and client access API.

Security Risks

This section describes some of the security risks involved with running EJB applications on the Internet, and what steps you can take to minimize the risks:

- When a client submits passwords using AM_SHAREDPASSWORD or AM_SHAREDPASSWORD_DB, the password is sent in the clear over the network. You should transmit passwords in this manner only if you are running over a trusted intranet. If you need to transmit passwords in this manner over the Internet, you should use IIOP / SSL, which provides encryption for confidentiality and integrity. To use IIOP / SSL, see "Security Risks" on page 5-27.
- If you are transmitting passwords using AM_SHAREDPASSWORD_MD_5, the password is encrypted, but it is still susceptible to being sniffed and reused. You should consider using IIOP / SSL even when using AM_SHAREDPASSWORD_MD_5.
- There is no method level access control. When you write your EJB components, you should check that private methods are not exposed to clients, that objects or methods that check the client's authorization are not bypassed, and that object references to EJB components are not given to unauthorized clients.

6

Oracle Wallet Manager

The Oracle Wallet Manager is a Java-based application that security administrators use to manage public-key security credentials on clients and server.

Overview

Public-key cryptography requires that parties who want to communicate in a secure manner possess certain security credentials. This collection of security credentials is stored in a wallet. A wallet is an abstraction used to store and manage security credentials for the client or the server.

Security credentials consist of a public/private key pair, a certificate, and trustpoints. Each entity that participates in a public key system must own a public/private key pair. The public key for an entity is published, so that other entities that want to send it secure information can encrypt that information with the recipient entity's public key. The Oracle Wallet Manager generates public/private key pairs for clients and servers.

A certificate authority (CA) issues a public key certificate. Certificates contain a unique serial number assigned to it by the CA, an algorithm identifier that identifies which algorithm was used to sign that certificate, the name of the CA that issued that certificate, a pair of dates during which the certificate is valid, the name of the certificate user, the entity's public key information, and the CA's signature.

A trustpoint is a third party identity that is qualified with a level of trust. A client or a server uses a trustpoint to validate that an entity is who it claims to be. Typically, the certificate authorities you trust are called trustpoints.

Clients and servers use these credentials to access secure services using public key cryptography. A wallet also represents a storage facility that is location and type transparent once it is opened. The wallet resource locator (WRL) provides all the necessary information to locate the wallet.

Purpose of the Oracle Wallet Manager

The Oracle Wallet Manager is used for the following tasks.

- To generate a public-private key pair and create a certificate request for submission to a certificate authority (CA).
- To install a certificate for the identity.
- To configure trustpoints for the identity.

Security Concepts

Following is a list of general security concepts and their associated definitions that you need to know.

Table 6–1 Security concepts

Term	Definition
Authentication	The recipient of an authenticated message can be certain of the message's origin (its sender). Authentication reduces the possibility that another person has impersonated the sender of the message.
Authorization	The set of privileges available to an authenticated entity.
Certificate	A certificate is created when an entity's public key is signed by a trusted identity: a certificate authority. This certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.

Table 6–1 Security concepts

Term	Definition
Certificate Authority	A trusted third party that vouches for the identity of an individual, company, or server and signs a certificate. For example, Verisign, Inc. is such an authority.
Confidentiality	A function of cryptography. Confidentiality guarantees that only the intended recipient(s) of a message can view the message (decrypt the ciphertext).
Cryptography	The act of writing and deciphering secret code resulting in secure messages.
Decryption	The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).
Digital Signature	A digital signature is created when a public key algorithm is used to sign the sender's message with the sender's private key. The digital signature assures that the document is authentic, has not been forged by another entity, has not been altered, and cannot be repudiated by the sender.
Encryption	The process of disguising the contents of a message and rendering it unreadable (ciphertext) to anyone but the intended recipient.
Identity	A user who is typically certified.
Integrity	The guarantee that the contents of the message received were not altered from the contents of the original message sent.
Non-repudiation	Undeniable proof of the origin, delivery, submission, or transmission of a message.
Public-Key Encryption	The process where the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using the recipient's private key.

Table 6–1 Security concepts

Term	Definition
Public/Private Key Pair	A mathematically related set of two numbers where one is called the private key and the other is called the public key. Public keys are typically made widely available, while a private key is available only to the owner. Data encrypted with a public key can be decrypted with its associated private key and vice versa. However, data encrypted with a public key cannot be decrypted with the same public key.
Trustpoint	A trustpoint is a third party identity that is qualified with a level of trust. The trustpoint is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust are called trustpoints.
Wallet	A wallet is an abstraction used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A wallet resource locator (WRL) provides all the necessary information to locate the wallet.
Wallet Resource Locator	A directory path that provides all the necessary information to locate a particular wallet.
WRL	See Wallet Resource Locator.
X.509	The public keys can be signed in various data formats. The X.509 format from ISO is one such popular format.

Using the Oracle Wallet Manager

Following is a list of Oracle Wallet Manager high level functions and their associated tasks.

- Starting the Oracle Wallet Manager
- Managing Wallets
  - Open an Existing Wallet
  - Create a New Wallet
- Managing Trustpoints
  - Add a New Trustpoint

- View Existing Trustpoint Information
- Delete a Trustpoint
- Save a Wallet to an Existing WRL (Wallet Resource Locator)

These functions and tasks are described in the remainder of this chapter.

Starting the Oracle Wallet Manager

The way you invoke the Oracle Wallet Manager depends on what kind of system you use.

Windows NT

Start the Oracle Wallet Manager on a Windows NT by clicking [Start] | Programs | Oracle Wallet Manager | Oracle Wallet Manager. You can also type **wmtgui** at a command prompt to start the application.

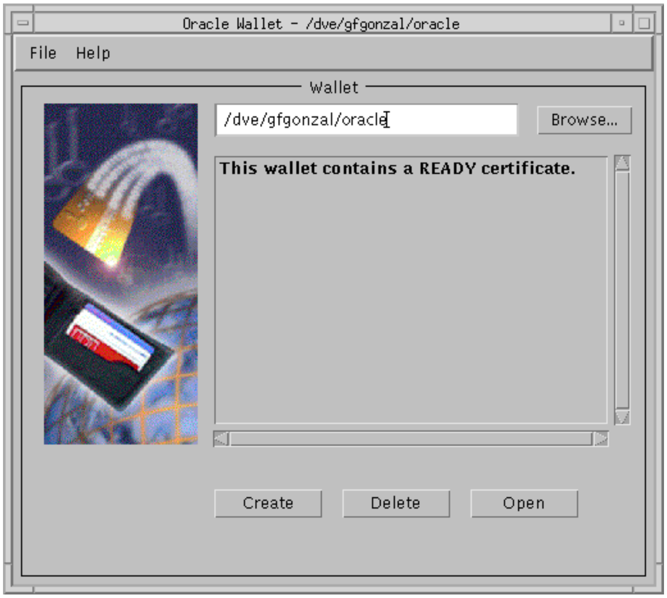
The Oracle Wallet Manager Wallet window is displayed. See Figure 6–1, “Oracle Wallet Manager”.

Solaris

Type **wmtgui** at the command line to invoke the Oracle Wallet Manager. The **wmtgui** utility is in the **$ORACLE_HOME/../wmt/bin** directory. Make sure this directory is in your PATH.

The Oracle Wallet Manager Wallet window is displayed. See Figure 6–1, “Oracle Wallet Manager”.

Figure 6–1 Oracle Wallet Manager



This wallet contains a READY certificate.

This window displays the default wallet location, the version of the certificate that is stored in the wallet, and the status of the wallet: EMPTY, REQUESTED, or READY. The following tables describe the fields and buttons in the window:

Table 6–2 Wallet location description

Field Name	Description
Wallet Location	Display the default wallet file location. Click the Browse... button to locate a wallet at another location.
Wallet Information	Display the version of the Oracle Wallet Manager that you are using and the status of the certificate (if any) installed in the wallet.

Table 6–3 Wallet location description

Button Name	Function
Create	Create a new wallet.

Table 6–3 Wallet location description

Button Name	Function
Delete	Delete the wallet displayed in this window.
Open	Open the wallet displayed in this window.

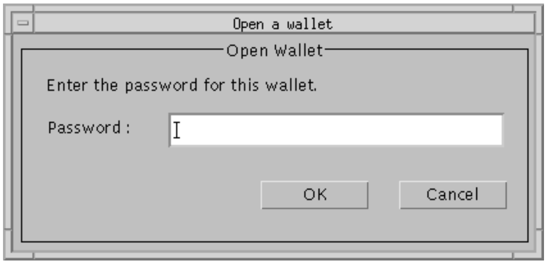
Managing Wallets

Use the Oracle Wallet Manager to open, view, or modify an existing wallet or to create a new wallet.

Open an Existing Wallet

The Oracle Wallet Manager enables wallet owners to open their default wallets. The default wallet is displayed in the Oracle Wallet Manager Start-up window. Wallet owners must provide a valid wallet resource locator (WRL) and the correct password to open the wallet. Refer to Figure 6–2, "Open the Default wallet".

Figure 6–2 Open the Default wallet



Open a wallet

Open Wallet

Enter the password for this wallet.

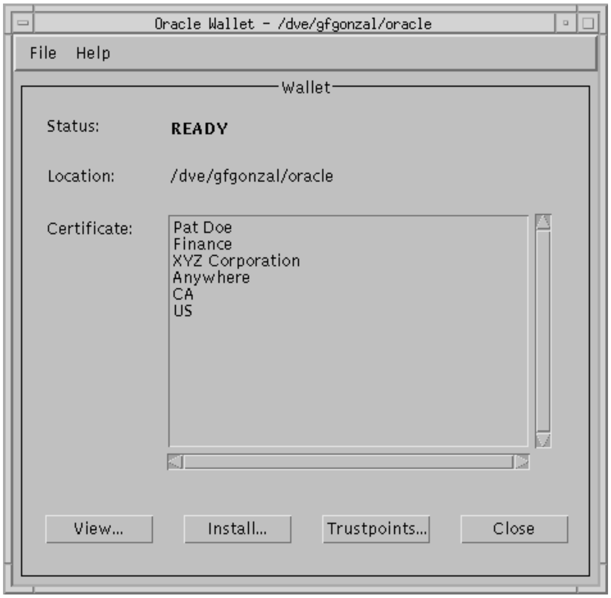
Password :

1. Click Open on the Oracle Wallet Manager start-up window.
   The Open Wallet Password window appears.

2. Type your password. Click Cancel to return to the Oracle Wallet Manager start-up window, or click OK to continue.
   The Wallet window appears. See Figure 6–3, "Default wallet".

Note: An error window titled "Failed to Open wallet!" will appear after you type an incorrect password. Click OK to return to the Oracle Wallet Manager Start-up window. Check your password and try again.

Figure 6–3 Default wallet



Oracle Wallet - /dve/gfgonzal/oracle

Status: READY

Location: /dve/gfgonzal/oracle

Certificate:
Pat Doe
Finance
XYZ Corporation
Anywhere
CA
US

View the Wallet Contents

Use the Wallet window to access functions that allow you to view or modify the wallet's contents. This window contains the following fields and buttons:

Table 6–4 Wallet contents description

Field Name	Description
Status	This field displays the status of the wallet. The three values are EMPTY, REQUESTED, and READY. A status of EMPTY means that no certificate is requested or installed in the wallet. A status of REQUESTED indicates that the certificate request for your wallet has been generated. A status of READY means that you have a certificate.
Location	The directory in which the wallet is stored.
Certificate	The name of the identity for whom this certificate is installed in the wallet.

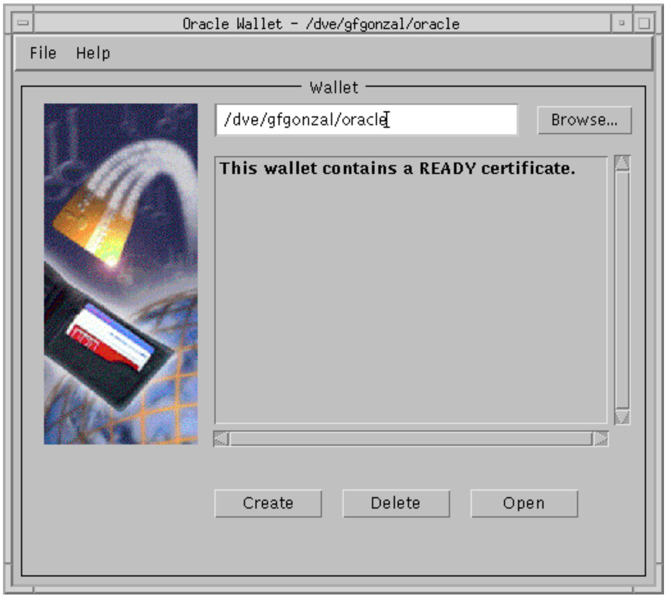
Table 6–5 Wallet contents description

Button Name	Function
View	View the installed certificate.
Install	Install a new certificate into the wallet.
Trustpoints	View and manage the trustpoints installed in your wallet.
Close	Return to the Oracle Wallet Manager start-up window.

Create a New Wallet

Follow the steps below to create a new wallet. The steps assume that you have started the Oracle Wallet Manager and are at the program's initial window. See Figure 6–4, "Oracle Wallet Manager Start-up Window".

Figure 6–4 Oracle Wallet Manager Start-up Window



/dve/gfgonzal/oracle

This wallet contains a READY certificate.

This window contains the following fields and buttons:

Table 6–6 Creating a new wallet

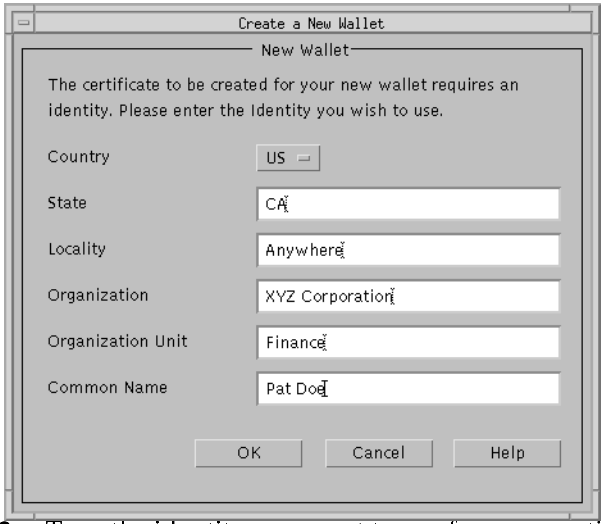
Field Name	Description
Wallet Location	Display the default wallet file location. Click the Browse... button to locate a wallet at another location.
Wallet Information	Display the version of the Oracle Wallet Manager that you are using and the status of the certificate (if any) installed in the wallet.

Table 6–7 Creating a new wallet

Button Name	Function
Create	Create a new wallet.
Delete	Delete the wallet displayed in this window.
Open	Open the wallet displayed in this window.

1. Click Create to create a new wallet.
   The New Wallet Identity window appears.

Figure 6–5 New wallet identity



Create a New Wallet

New Wallet

The certificate to be created for your new wallet requires an identity. Please enter the Identity you wish to use.

Country: US

State: CA

Locality: Anywhere

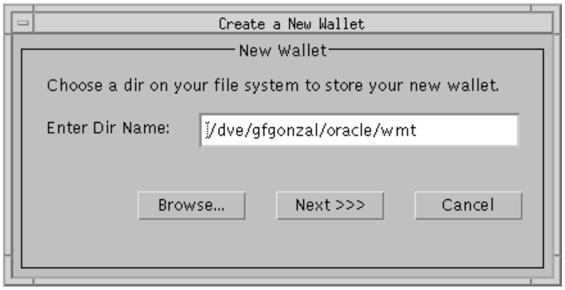
Organization: XYZ Corporation

Organization Unit: Finance

Common Name: Pat Doe

2. Type the identity you want to use for your certificate, and click OK.
   The New Wallet location window appears prompting you to choose a directory on your file system in which to store the new wallet.

Figure 6–6 Specify Wallet File Location



Create a New Wallet

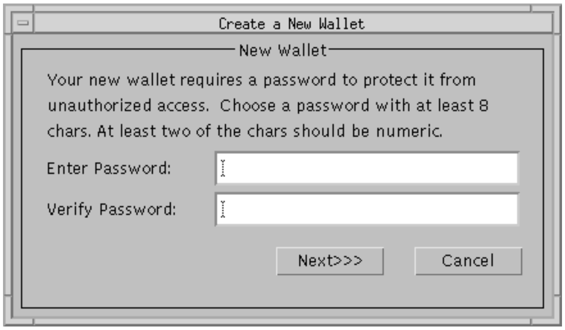
New Wallet

Choose a dir on your file system to store your new wallet.

Enter Dir Name: /dve/gfgonzal/oracle/wmt

3. Type the file location for the new wallet. Click Browse to find a wallet in another directory. Click Cancel to return to the initial program window, or click Next to continue.
   The New Wallet password window appears.

Figure 6–7 Type a password for the New Wallet



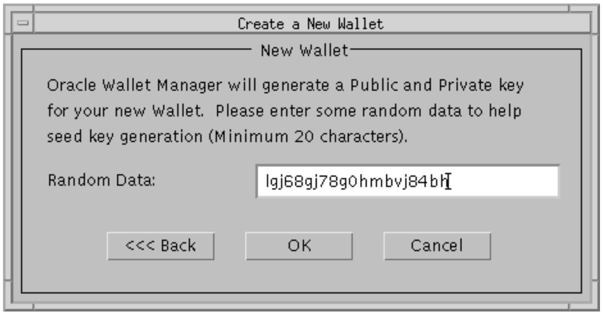
Your new wallet requires a password to protect it from unauthorized access. Choose a password with at least 8 chars. At least two of the chars should be numeric.

4. Type your password once in the Enter Password field, and re-type that same password in the Verify Password field.

> **Note:** An Error window appears if the two passwords you typed do not match. Click OK to return to the New Wallet password window, and repeat step 4 above.

5. Click Next to continue.
   The New Wallet Random Data window appears.

Figure 6–8 New Wallet random data

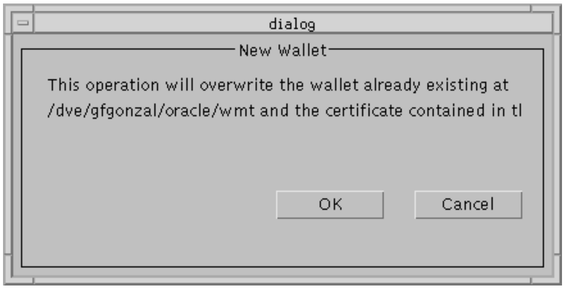


Oracle Wallet Manager will generate a Public and Private key for your new Wallet. Please enter some random data to help seed key generation (Minimum 20 characters).

Random Data: lgj68gj78g0hmbvj84bh

6. Type a random string of at least 20 characters in length in the field. This character string will be used to seed the public/private key generation, and click OK. See Figure 6–8, “New Wallet random data”.
   A New Wallet dialog box appears.

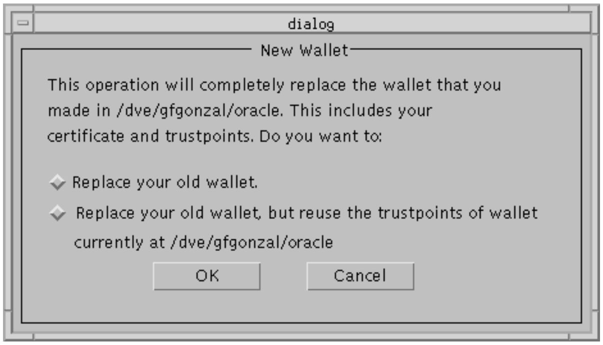
Figure 6–9 Create a New Wallet



This operation will overwrite the wallet already existing at /dve/gfgonzal/oracle/wmt and the certificate contained in tl

7. The dialog box informs you that the new wallet will overwrite the wallet, certificate and trustpoints that already exist at the default file location. This occurs when you already have an existing wallet in the default location. Click OK. The Replace Wallet dialog box appears. See Figure 6–10, “Replace Wallet dialog box”.

Figure 6–10 Replace Wallet dialog box



This operation will completely replace the wallet that you made in /dve/gfgonzal/oracle. This includes your certificate and trustpoints. Do you want to:

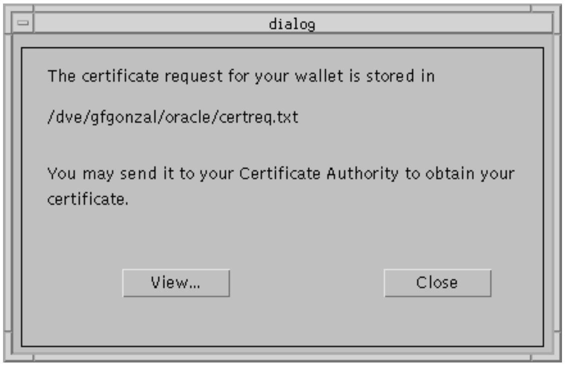
◇ Replace your old wallet.

◇ Replace your old wallet, but reuse the trustpoints of wallet currently at /dve/gfgonzal/oracle

8. Click the Replace your old wallet radio button to replace the entire wallet including the certificate and trustpoints, or click the Replace your old wallet, but reuse... radio button to replace the wallet but reuse the old wallet’s trustpoints. Click Cancel to return to the initial program window, or click OK to continue.

9. A dialog appears displaying the location of your certificate request file: certreq.txt. See Figure 6–11, “Certificate request location”. This is the file you use to send to your Certificate Authority.

Figure 6–11 Certificate request location



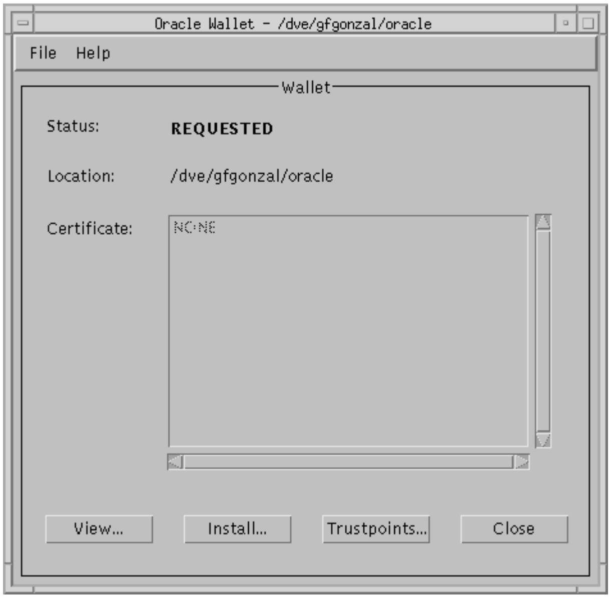
The certificate request for your wallet is stored in

/dve/gfgonzal/oracle/certreq.txt

You may send it to your Certificate Authority to obtain your certificate.

10. Click View to view your certificate request, or click Close to close this window.
11. The Wallet window appears with a status value of REQUESTED and a certificate value of NONE. See Figure 6–12, “New wallet with a requested certificate”.

Figure 6–12 New wallet with a requested certificate



Status: **REQUESTED**

Location: /dve/gfgonzal/oracle

Certificate: NONE

Install a Certificate into the New Wallet

Once you send the certificate request to the certificate authority, wait until you receive an e-mail reply containing your signed certificate. Depending upon the Certificate Authority, you may receive a certificate file such as **certificate.txt**. Proceed to install the certificate into the new wallet as follows. Option 1, below, describes how to install a certificate from a file, and Option 2, below, describes how to install a certificate from the contents of an e-mail message from the CA.

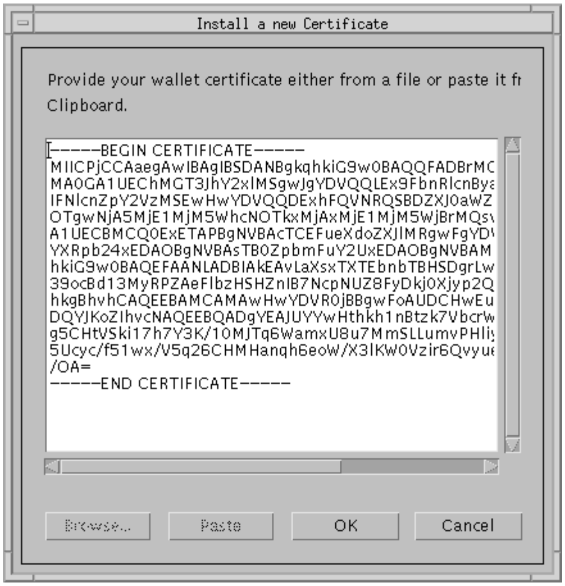
Option 1: Install a Certificate from a File

1. Open the e-mail from your certificate authority and locate the certificate text. The certificate content is usually delimited by the words BEGIN CERTIFICATE and END CERTIFICATE.

2. Click Install on the Wallet window. See Figure 6–12, “New wallet with a requested certificate”. The Install a new Certificate window appears.

3. Click Browse. A directory window appears. Use this window to locate the certificate.txt file (it may also have some other name depending upon the Certificate Authority). Click the file name to select it, and click Open. The contents of the certificate file will appear in the window area. See Figure 6–13, “Certificate text pasted into window”.

Figure 6–13 Certificate text pasted into window



Provide your wallet certificate either from a file or paste it fr
Clipboard.

```
------BEGIN CERTIFICATE------
MIICPjCCAaegAwIBAgIBSDANBgkqhkiG9w0BAQQFADBrMC
MA0GA1UEChMGT3JhY2xlMSgwJgYDVQQLEx9FbnRlcnBy
IFNlcnZpY2VzMSEwHwYDVQQDExhFQVNRQSBDZXJ0aWZ
OTgwNjA5MjE1MjM5WhcNOTkxMjAxMjE1MjM5WjBrMQs
A1UECBMCQ0ExETAPBgNVBAcTCEFueXdoZXJlMRgwFgYD
YXRpb24xEDAOBgNVBAsTB0ZpbmFuY2UxEDAOBgNVBAM
hkiG9w0BAQEFAANLADBIAkEAvLaXsxTXTEbnbTBHSDgrLw
39ocBd13MyRPZAeFlbzHSHZnIB7NcpNUZ8FyDkj0Xjyp2Q
hkgBhvhCAQEEBAMCAMAwHwYDVR0jBBgwFoAUDCHwEu
DQYJKoZIhvcNAQEEBQADgYEAJUYYwHthkh1nBtzk7VbcrW
g5CHtVSki17h7Y3K/10MJTq6WamxU8u7MmSLLumvPHli
5Ucyc/f51wx/V5q26CHMHanqh6eoW/X3lKW0Vzir6Qvyu
/OA=
------END CERTIFICATE------
```

4. Click OK.
   You are returned to the Wallet window. Its status changes to READY.

Option 2: Install a Certificate from the Body of an E-mail

1. Open the e-mail you received from the certificate authority.
2. Highlight and copy the certificate text from the body of the e-mail.
3. Click Paste on the Install a new Certificate window. The certificate text will be pasted into this window. See Figure 6–13, “Certificate text pasted into window”.
4. Click OK.
5. You are returned to the Wallet window. Its status changes to READY.

Managing Trustpoints

A trustpoint is a third party identity contained within a wallet that is qualified with a level of trust. The trustpoint is used when an identity is being validated as the entity it claims to be. Trustpoints are also referred to as Trusted CAs (certificate authorities). Use the Oracle Wallet Manager to manage the trustpoints in your wallet. You can add a new trustpoint, view existing trustpoint information, and delete a trustpoint. A default set of four trustpoints is installed in your default wallet when you install the Oracle Wallet Manager. See Figure 6–14, “Trustpoints”.

The Oracle Wallet Manager installation installs a default wallet for you. This default wallet contains a list of Verisign trustpoints only. Thus, any certificate issued by a CA other than Verisign is not trusted by default. To install a certificate issued by any other CA (other than Verisign), you will need to get the certificate and the corresponding private keys from this CA. You must have the CA’s certificate to be used as a trustpoint. Thus, your wallet will be complete once it has the following three files:

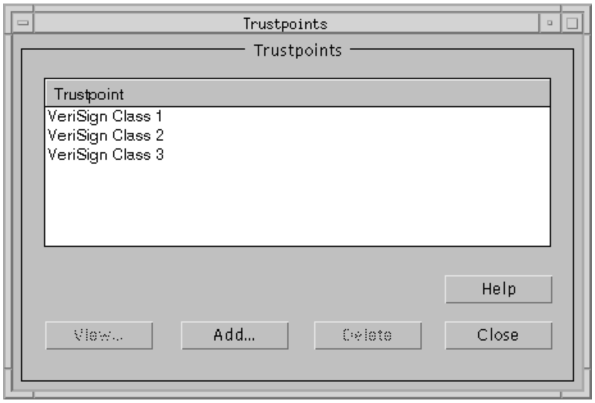
- certificate
- private key
- trustpoint

Add a New Trustpoint

Add a new trustpoint to your wallet as follows.

1. Click Trustpoints on the Wallet window.
   The Trustpoints window appears. See Figure 6–14, “Trustpoints”.

Figure 6–14 Trustpoints

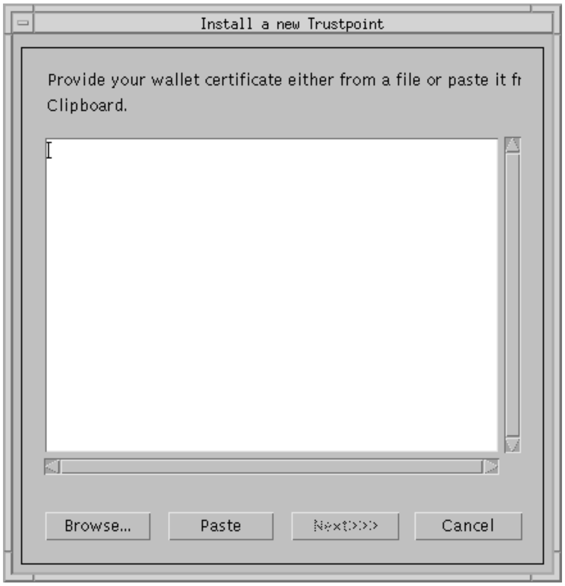


Trustpoint
VeriSign Class 1
VeriSign Class 2
VeriSign Class 3

2. Click Add.

   The Install a New Trustpoint window appears.

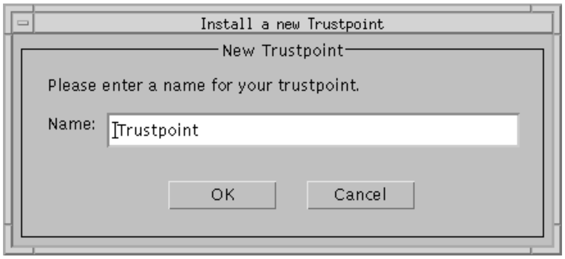
Figure 6–15 Install a new Trustpoint



Provide your wallet certificate either from a file or paste it fr
Clipboard.

3. This is the window where you will paste the trustpoint certificate into. Click Paste.
4. The certificate text appears in the body of the window. Click Next.
   The Trustpoint Name window appears.

Figure 6–16 Trustpoint Name window



Please enter a name for your trustpoint.

Name: Trustpoint

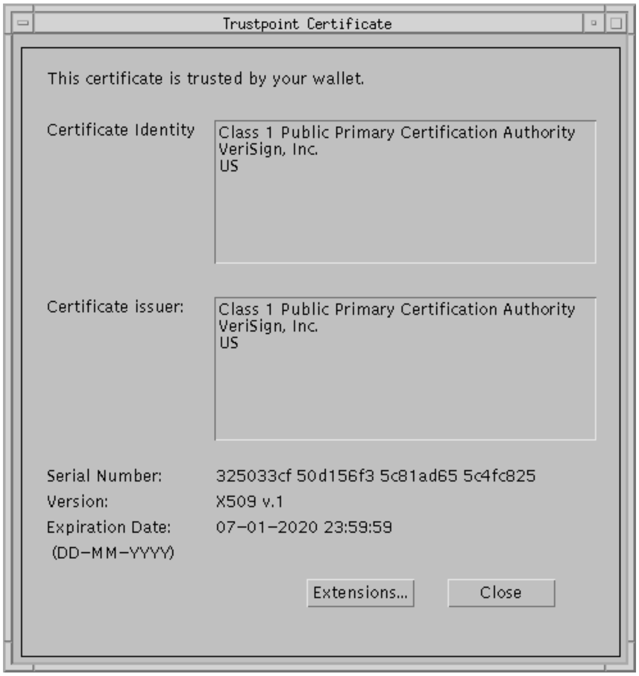
5. Type a name for the trustpoint alias. This name can be any set of alphanumeric characters, but it cannot contain any spaces.
6. Click Cancel to return to the previous window, or click Next to continue.
   The Trustpoint you created is added to the list of trustpoints in the Trustpoints window.
7. Click Close, and you are returned to the Wallet window.

View Existing Trustpoint Information

You can view detailed trustpoint information from the Trustpoints window as follows.

1. Click the name of the trustpoint for which you want to view detailed information.
2. Click View.
   The Trustpoint Certificate window appears. See Figure 6–17, “Trustpoint Certificate”.

Figure 6–17 Trustpoint Certificate



Trustpoint Certificate

This certificate is trusted by your wallet.

Certificate Identity: Class 1 Public Primary Certification Authority
VeriSign, Inc.
US

Certificate issuer: Class 1 Public Primary Certification Authority
VeriSign, Inc.
US

Serial Number: 325033cf 50d156f3 5c81ad65 5c4fc825
Version: X509 v.1
Expiration Date: (DD–MM–YYYY) 07–01–2020 23:59:59

3. Review the trustpoint certificate information that was installed into your wallet. This information includes the certificate identity and the certificate issuer.
4. Click Extensions to display the X.509 v3 certificate extension information for your wallet trustpoint, or click Close to return to the Trustpoints window.

Delete a Trustpoint

The Oracle Wallet Manager offers you the option of deleting selected trustpoints in the event that they become compromised. Delete a Trustpoint from the Trustpoints window as follows.

1. Click the name of the trustpoint listed in the Trustpoint column.
2. Click Delete.
3. A dialog box appears prompting you with, “Do you really want to delete this trustpoint?”

4. Click Yes to delete the trustpoint. You are returned to the Trustpoints window, and the deleted trustpoint is no longer displayed in the Trustpoint list.
5. If you had clicked No, you are returned to the Trustpoints window, and the trustpoint remains displayed in the Trustpoint list.
6. Click Close to return to the Wallet window.

Note: Click File > Save in the Wallet window to save your wallet and for the changes to take effect.

Save a Wallet to an Existing WRL (Wallet Resource Locator)

Click File > Save to save changes you make to the wallet.

Index